Solar Photovoltaic IV Curve Tracer

by

Jonathan Skelly

Senior Project

Advisor: Professor Dale Dolan, Cal Poly EE Faculty

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University

San Luis Obispo

June 2021

Table of Contents

List of Tables and Figures

Abstract

The Solar Photovoltaic Current-Voltage (IV) Curve Tracer, sponsored by Cal Poly Professor Dale Dolan, characterizes solar array current vs. voltage curves for any given temperature and irradiance. The curve tracer is battery powered, and functions autonomously across loading conditions from short circuit to open load on any sub-450-watt solar array. The curve tracer supports separate data logging modes for both single cells and single modules to maximize data accuracy for each. For testing full strings of solar panels, up to 10 IV curves may be stored in persistent EEPROM memory. Stored data may be recalled later for output to a personal computer in comma separated value format.

This completed project is designed to teach Cal Poly students about the operation and continued maintenance of any solar array operating within the previously outlined specs during related coursework. This project will allow students to observe the optimal operating point of solar arrays at any temperature, illuminance, and load to identify problems due to mismatch, shading, soiling, and other functional errors. In a laboratory setting, extreme conditions may be tested and characterized to ensure a panel provides enough power and functions at all expected limits

The completed project could successfully characterize IV curves for single cells, record temperature, store 10 data sets, and output data in .csv format to a personal computer, in conjunction with working LCD and button interfaces. The irradiance sensor chosen was unfortunately not compatible with the chosen microcontroller, and Covid distancing gave little chance to test the device on full panels during the design stages. When the final project was tested on a full panel, it was not put in the correct mode for the high panel voltage, and the microcontroller was shorted. Future improvements would include overvoltage protection and a professionally manufactured PCB and chassis.

Chapter 1.
Introduction

Solar arrays are highly susceptible to changes in power generation over time. On average, panel power output drops by an average of 0.8% per year through age and use, leading to large losses and high mismatch between panels [1]. Short-term environmental effects also reduce panel efficiency, with certain particulate soiling creating up to 98% losses [2]. As solar cells are connected in series, sharing a single current line between each string, a single underperforming or broken cell can be a detriment to the entire power generating process. To combat these problems, frequent monitoring of panel performance is required to identify failed modules, mitigate environmental effects, and anticipate rate of degradation. Solar IV curve tracers are frequently chosen for this function.

Solar IV curve tracers usually feature an electronic load placed across a panel's terminals, which is swept through various loading conditions via a microcontroller [3]. Current and voltage (I-V) measurements are digitally recorded at each loading value, as well as ambient temperature and irradiance conditions to properly characterize the resulting data [3]. The data can then be graphed and analyzed for comparison against expected or nominal operation. Curve tracers frequently feature high-end features like embedded screens and wireless data transfer, regularly driving costs up to over $5000, well out of the price range of college students or fledgling solar engineers [4], [5].

The goal of this project is to develop an inexpensive alternative Solar IV Curve Tracer for personal or student use in solar array performance characterization. The device will aid in solar array design and maintenance by giving data-based insight to optimal operating points and actual generated power. By using collected IV data to determine the max operating power point, cells and modules can be appropriately loaded and connected to achieve desired outputs in the design phase. Existing solar arrays, given reduction in performance over time, can also be properly characterized using an IV curve tracer to note and diagnose power drops.

Chapter 2.
Customer Needs, Requirements, & Specifications

Customer Needs Assessment

Customer needs are each derived from the given project requirements, as well as similar curve tracers currently available.

1. The customer needs the device to be easy to operate. The product should be usable by students and inexperienced solar owners without a strong background in electronics or solar. Complex inputs, modes, or setup would reduce the userbase and impede usability.

2. The customer needs the device to be versatile. The product is not very useful if it can only be used under very limited conditions, and must be marketable to a wide range of needs.

3. The customer needs the product to be fast enough to be able to test many cells or panels in a reasonable amount of time, and so environmental conditions do not change between the beginning and end of the test. If it takes too long to form an IV curve, it is impossible to make efficient use of the device.

4. The customer needs the product to provide useable data. Abstract or insufficient data logged and presented would be no use to the customer. Quantitative data logs would allow interpretation and utilization of the data elsewhere.

5. The customer needs the project to be portable. If the project is stationary or too heavy, it would present difficulty in measuring separate panels, diminishing the usefulness of the design.

Requirements and Specifications

Each design specification is defined in order to meet all customer needs.

1. The design must measure temperature (0-80ºC) and irradiance (0-1000 W/m^2). Temperature and irradiance both highly affect solar panel operation and must be considered for accurate data. 80ºC was chosen due to it being safely above the highest recorded temperature on Earth (56.7 ºC), guaranteeing accurate functionality in realistic weather conditions while allowing room for the panel temperature to be hotter than ambient [6]. 1000 W/m^2 is considered the irradiance under full sunlight [7].

2. The design must supply a variable load from short circuit to open load, as defined by the project outline. This will allow the panel under test to be subject to the full range of possible load scenarios, allowing for a wide range of test data.

3. The design must be capable of testing a single cell (0-0.7V/0-12A) and a single module (0-90V/0-12A), as well as a max power point of 450W, defined by the project outline. This will allow the device to test a wide range of solar arrays, with limits defined by the cell and module electrical characteristics. 12 amperes is the current limit for each because it is well above the standard short circuit current of most panels (~10A) [7]. 0.7 and 90 volts are chosen as the max cell and module voltages based on usual panel electrical characteristics [7].

4. The device must feature a basic user interface, including a character-based LED display, numerical buttons, an enter button, and a back button. These will be used for measurement setup, confirmation, and data storage.

5. The device must be capable of automatically taking 1000 data points in a single curve. Around 1000 data points is the standard resolution when testing maximum voltage of similar products [4], [5].

6. The device must log a full curve in under 30 seconds. This short timeframe will ensure the product is fast and useful, while also reducing the risk that lighting and weather conditions have changed between the beginning and end of the test.

7. The device must be capable of outputting to a .csv file via USB on a personal computer for displaying logged data upon completion of a curve or upon recalling stored data. This will allow use of the visual data analysis function of similar products, while saving cost on high-resolution on-board displays and proprietary computer analysis programs.

8. The device must be under 10 pounds, so it may be easily used and transported. This weight is based on that of similar products, with a higher upper bound given for potential fabricated chassis weight [4], [5].

9. The device must run on 9V battery power to keep the product portable and the necessary batteries easily obtainable. 9V keeps the number of batteries reasonable and provides plenty of current when regulated down to common 3.3 or 5V chip supplies [9].

10. The device must have a plastic enclosure, with access to the main circuitry for maintenance, a compartment for batteries, and holes where appropriate for panel leads and USB.

Table I identifies how each described engineering specification satisfies the outlined needs of the customer.

TABLE I

SOLAR IV CURVE TRACER SPECIFICATIONS

Marketing Requirements	Engineering Specifications	Justification
2	Take temperature and irradiance as parameters.	Factoring in temp and irradiance allows accurate use in many types of environments.
2,4	Supply a load that is continuously variable between short circuit and open load.	A wide load range allows testing of many different scenarios, and makes the presented IV curve cover a greater, and more useful, range.
2	Capable of testing a single cell (0-0.7V/0-12A) and a single module (0-90V/0-12A) at a maximum of 450W.	The ability to test cells and modules allows the customer to decide what they would like to use the product for, and grants a wide range of use.
1	Include a basic user interface.	This will allow the user to easily save data, receive feedback, and make operation-based decisions.
1,4	Take 1000 data points per curve automatically.	1000 points will allow for a smooth and accurate curve, and is based on curve resolutions offered by similar products. Automatic data logging will allow a full curve to be characterized at a button press.
3	Log a full curve in under 30 seconds.	Keeping the timeframe for a full curve low will allow multiple tests to be taken in a reasonable time.
6	Weigh under 10 pounds.	Making the device light allows it to be carried and easily used on different projects.
4,6	Output logged data to a .csv file on a PC via USB.	Computer communication will allow for visualization of data. Data outputted to a .csv file can be copied into other documents or programs for analysis. USB connection allows for a portable device.
6	Must use 9V battery power.	No stationary power supply allows for portability.
1, 6	Must have plastic enclosure.	The device will be easier to transport, keep clean, and prevent from damage if it is enclosed.
Marketing Requirements 1. Easy to operate. 2. Versatile. 3. Fast performance. 4. Provides useable data 5. Portable		

Chapter 3.
Functional Decomposition

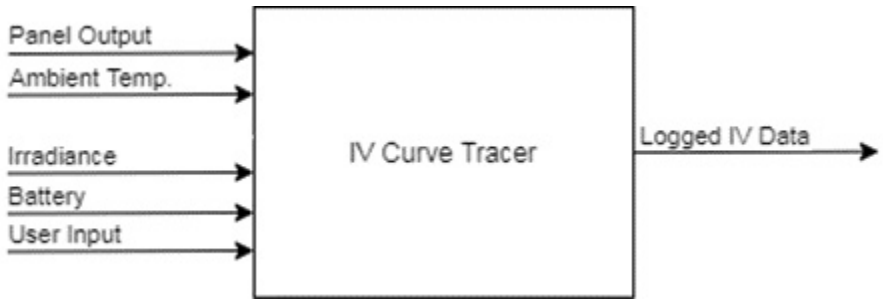
Block Diagrams

Level 0

To streamline functionality, the IV curve tracer is designed to take few active inputs from the operator. The system, at the highest level, requires an output from the connected panel, ambient temperature and irradiance measurements, supplied battery power, and minimal user input to function. The single resulting output is a logged IV curve. These inputs and outputs were derived from the engineering specifications in Chapter 1.

FIGURE I

LEVEL 0 BLOCK DIAGRAM



The diagram inputs and outputs are summarized in Table II below.

TABLE II

LEVEL 0 FUNCTIONAL TABLE

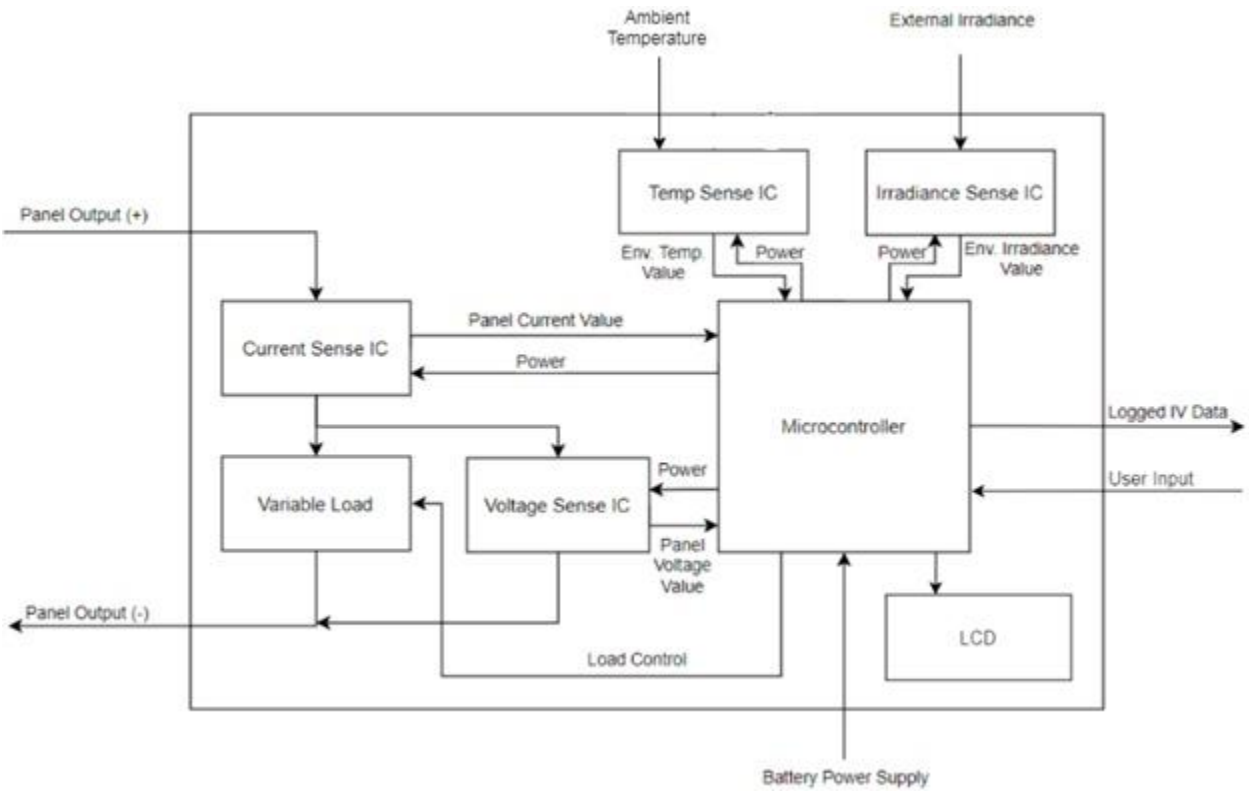
Module	Solar IV Tracer
Inputs	• Output terminals of the solar panel under test. • Temperature of the environment. • Irradiance of the environment. • Battery power source. • User input.
Outputs	• Serial bus with logged IV data.
Functionality	Logs IV curve of a panel under test from 0 to 450W, as well as open circuit. Mode and data point count are input by user. Reads ambient temperature and irradiance. Completes operation in under 30 seconds and sends the data to a computer.

Level 1

Breaking the design down further, the curve tracer will operate around a common MSP432 microcontroller from Texas Instruments [10]. The user will input the desired mode of operation, save file, or confirmation. The microncontroller will display these inputs, as well as starting/ending notifications, on the LCD. The micrcontroller will control a variable electronic load through ROHM SiC mosfets to sweep the panel output at the desired loading condition [11]. Integrated circuits will automatically perfrom each required measurment and send the data to the micrcontroller for storage. The micrcontroller will output all data to a computer source when data acquisition has finished. Figure II deomstrates these interconnections.

FIGURE II

LEVEL 1 BLOCK DIAGRAM



The inputs, outputs, and functionality of each module are detailed individually in Table III for clear functional decomposition.

TABLE III

LEVEL 1 FUNCTIONAL TABLE

Module	Microcontroller
Inputs	• Battery power source, regulated to +5V. • Collected solar panel voltage data • Collected solar panel current data • Collected temperature data • Collected irradiance data • User Input
Outputs	• Serial bus with logged IV data • IC voltages, if less than 5V • Load control • LCD data
Functionality	Uses current and voltage data sent by peripheral ICs to log and output IV data from internal memory. Uses temperature and irradiance levels to categorize data and make appropriate measurment adjustments. Powers all peripheral ICs with no more than +5V and +3.3V. Controls variable load, working from short circuit to max load (up to 450W) in less than 30 seconds. User input will be used to enter desired mode, confirmations, and choices. Current mode and status displayed on LCD.

Module	LCD
Inputs	• Microcontroller status inforation
Outputs	• None
Functionality	2-line LCD display. Provides user with visual confrimation of mode selected, number of test points selected, load value in manual mode, operation start, and operation end.

Module	Current Sense IC
Inputs	• Solar panel output terminals, series • VDD from micrcontroller, no more than +3.3V
Outputs	• Serial bus with panel current data
Functionality	Low-impedance IC that measures current through the supplied load, up to 12A. Sends data to micrcontroller 1000 times every 30 seconds.

Module	Voltage Sense IC
Inputs	• Solar panel output terminals, parallel • VDD from microntroller, no more than +3.3V
Outputs	• Serial bus with panel voltage data
Functionality	High-impedance (>1Mohm) IC that measures voltage across the supplied load, up to 90V. Sends data to micrcontroller 1000 times every 30 seconds.

Module	Temp Sense IC
Inputs	• External temperature up to 80ºC. • VDD from microntroller, no more than +5V
Outputs	• Serial bus with temperature data
Functionality	Measures environmental tempurature and panel temperature. Sends data once per operation to microcontroller, before IV data logging

Module	Irradiance Sense IC
Inputs	• External irradiance, 1000 w/m^2 max • VDD from microntroller, no more than +5V
Outputs	• Serial bus with irradiance data
Functionality	Measures environmental light. Sends data once per operation to microcontroller, before IV data logging

Module	Variable Load
Inputs	• Output terminals of the solar panel under test, series. • Microntroller settings.
Outputs	• Variable load, short circuit to open load
Functionality	Changes load value across solar panel terminals to allow voltage and current ICs to take data points, 1000 times per 30 seconds.

Chapter 4.

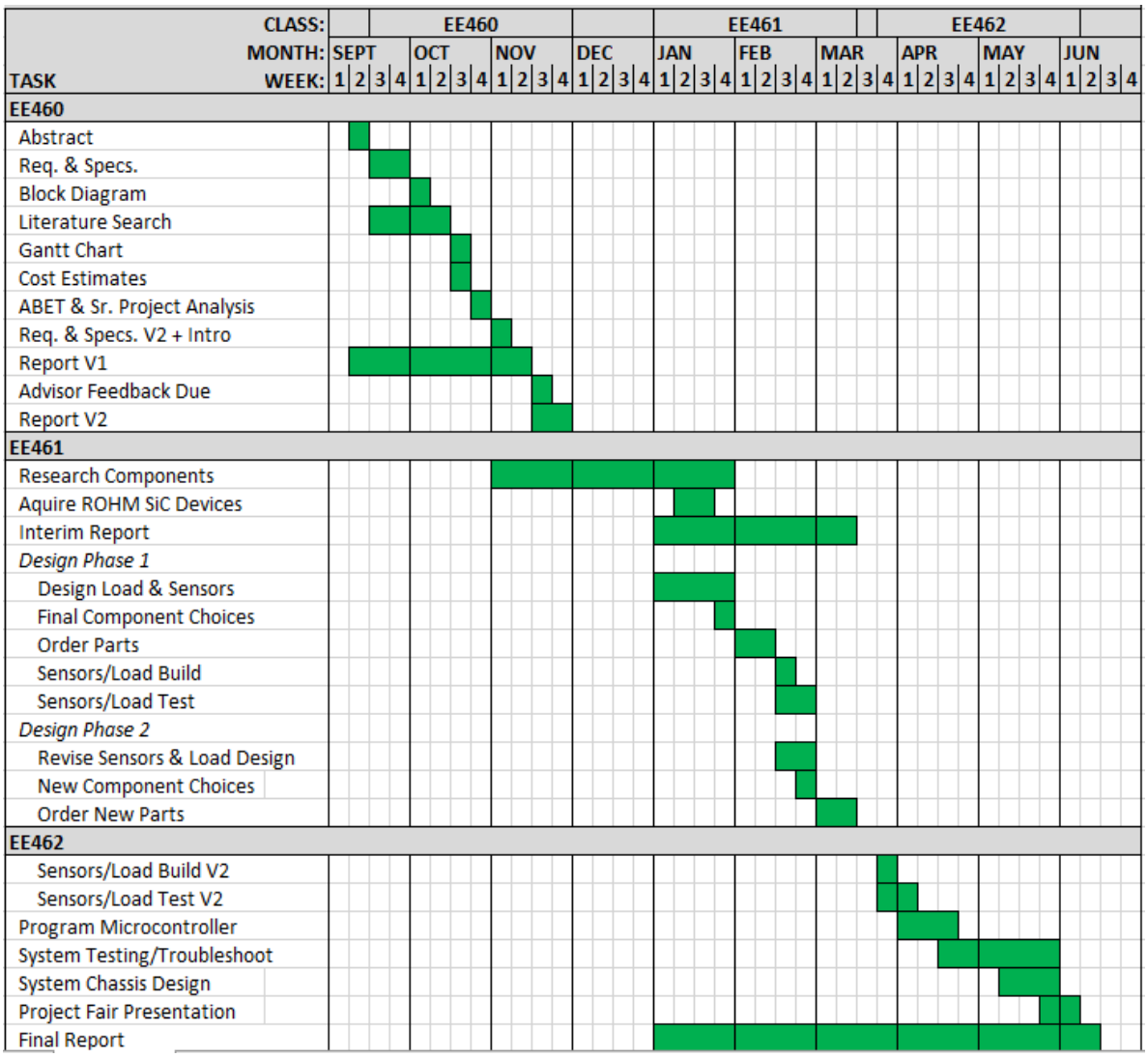
Project Planning

Gantt Chart

The project timeline in Figure III is primarily devised from university submission deadlines and estimated labor time. Priority was given to project design and report drafting when distributing time. Worst-case timeframes are assumed for each task.

FIGURE III

GANTT CHART



Cost Estimate

Component price estimates in Table IV are derived from median component cost on Digi-Key [9]. Other costs are estimates based on projected project needs. Labor cost is derived for informational purposes only; no monetary compensation is being accepted for this project.

TABLE IV

COST ESTIMATE

ITEM	Optimistic Cost	Likely Cost	Pessimistic Cost	Item Cost (Coulston)	Justification
Microcontroller	$0	$0	$75	$12.50	At best case and most likely, the TI MSP432 already owned will be sufficient. Worst case, a new microcontroller with higher voltage tolerance/expanded features would be needed.
ROHM SiC Devices	$0	$0	$15	$2.50	SiC devices will be provided by ROHM. In the case that they cannot be contacted or the devuces not delivered, purchasing replacement SiC parts will be necessary.
Monitoring Ics	$20	$60	$80	$56.67	Monitoring ICs are relatively inexpensive, most of the cost is shipping. High-power requirements will drive cost, as will re-purchasing in later design cycles.
LCD Display	0	14	20	$12.67	Best case, an LCD already owned can be utilized. Most likely, a larger model will be desired with more characters. Worst case, a higher-quality model is selected at a higher price.
Other Electrical Components	$12	$20	$40	$22.00	At the best case, many of the necessary components (resistors, capacitors, etc.) are already owned, most of the cost is shipping for remaining parts. Worst case, none are owned and all must be purchased. Most likely only a few new passive parts will be needed, along with a variety of transistors and other common parts.
Product Housing	$10	$20	$30	$20.00	Optimistically, a housing for the project can be chosen from an existing product, but more liekly one will have to be fabricated.
Labor Cost	$5,121	$10,243	20,486	$11,096.50	Considering all 6 class units (30 weeks) involved in the Electircal Engineering senior project, at the calculated average hourly earnings of an entry-level elctrical engineer [12]. Optimistic 1 hr/day, likely 2 hrs/day, pessimistic 4 hrs/day.
TOTAL	$5,163	$10,357	$20,746	$11,222.83	

Chapter 5.

Design

Microcontroller

An MSP432 launchpad from Texas instruments was chosen for familiarity, ease of programming, high number of output pins, and high likelihood that a Cal Poly student would have access to one given their requirement in other courses. The MSP will control the loads, liquid crystal display (LCD), and data storage, while receiving and processing data from the current, voltage, temperature, and irradiance sensors, as well as user input from the number pad. The MSP features both +5V and +3.3V supplies, defining the operating voltage of the following components. A simple LM340 linear voltage regulator will provide the +5V necessary for the microcontroller from the +9V battery supply [13].

The MSP432 features 24 onboard analog to digital converters (ADCs), that will be used to read the measurements from the monitoring ICs. The ADCs have a maximum input voltage of +3.3V, restricting the supply range of the sensors [10].

The MSP432 is capable of universal asynchronous receiver/transmitter (UART) serial communication via USB, which will be used to transfer logged data to a personal computer. Inter-integrated circuit (I2C) and serial peripheral interface (SPI) communication support also allows for a wider range of usable ICs and reduces the connections needed [13].

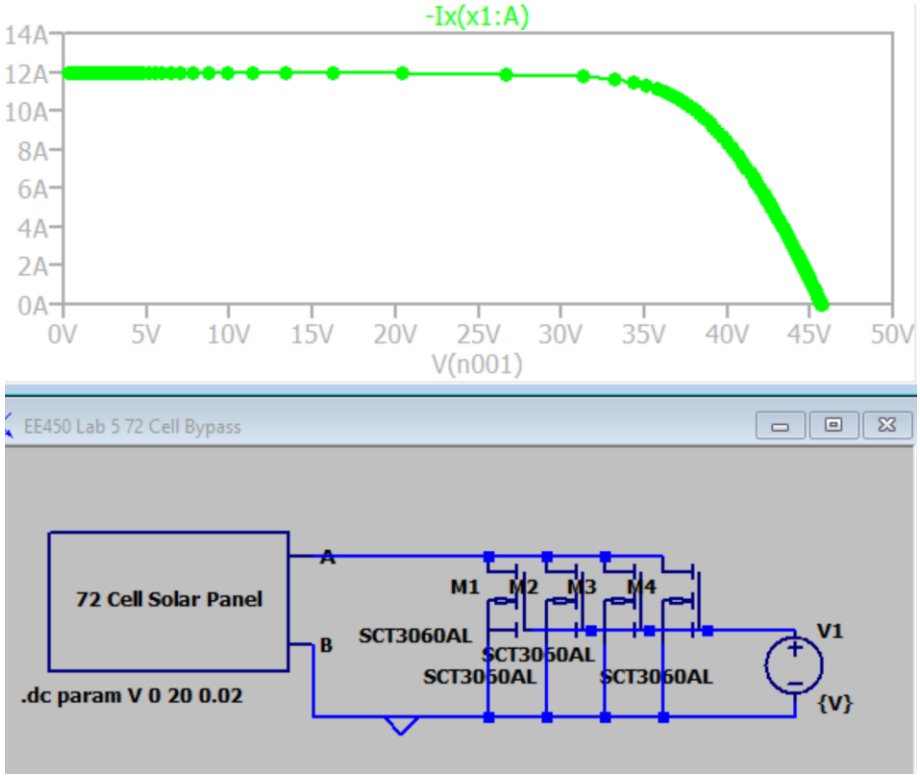
Electronic Load

The electronic load forms the main aspect of circuit design for the solar IV curve tracer. As per design requirements, it is necessary to use ROHM Semiconductor SiC mosfets. The SCT3060AL was chosen due to its balance of price versus functionality, exhibiting maximum characteristics of 650V and 12A – well above the design requirements of +90V/12A [11]. However, the maximum 165W mandates a reduction in the power dissipation required of each mosfet used in the electronic load. To meet the design requirement of handling 450W, four 3060AL's will be put in parallel between the high and low lines of the solar DUT. The gates of all four will be tied together to guarantee that each mosfet will be in the same on-state and equally share the load. In this configuration, the load can handle up to 660W, surpassing the design requirement.

LT Spice simulations of the parallel SiC mosfet design, shown in Figure IV, prove that they are fully capable of supplying the full load range necessary to complete a satisfactory solar IV curve. A 72-cell panel model and ROHM-provided SCT3060AL model are used for increased accuracy.

FIGURE IV

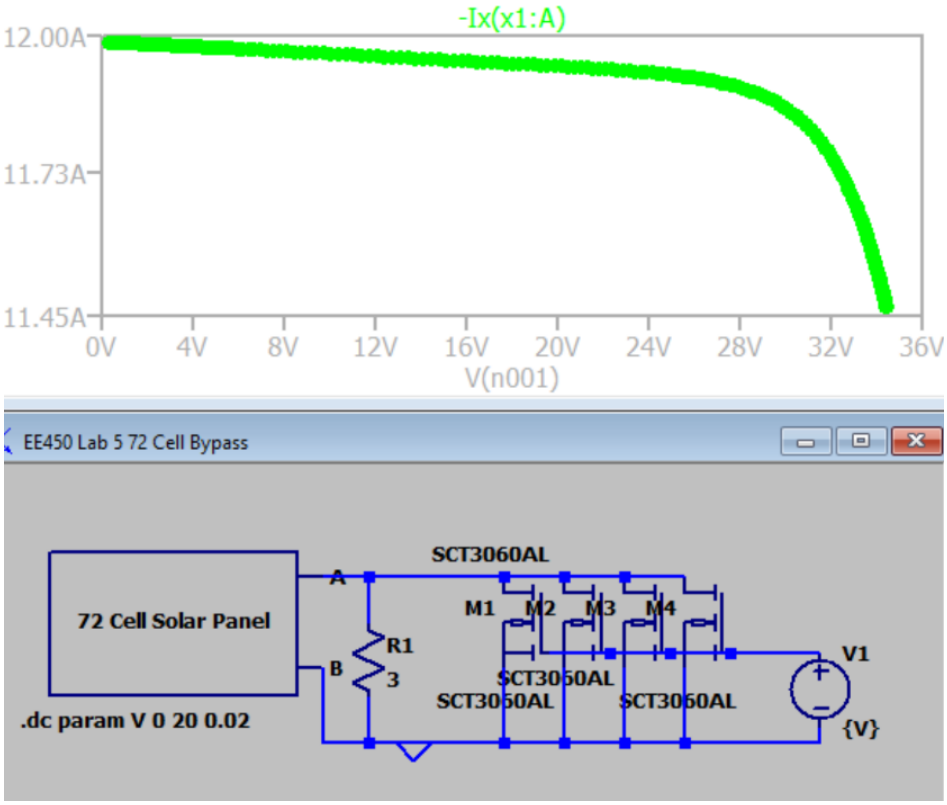
PANEL IV CURVE SIMULATION



Unfortunately, Figure IV shows that the transition region of the mosfets produce large resistance steps for every gate voltage step, resulting in a loss of resolution from around +10 to +35V. Further simulations similar to Figure V remedied this problem by placing a small resistor in parallel with the load, which grants better resolution in the aforementioned range at the cost of data near open load.

FIGURE V

PARALLEL RESISTOR PANEL IV CURVE SIMULATION

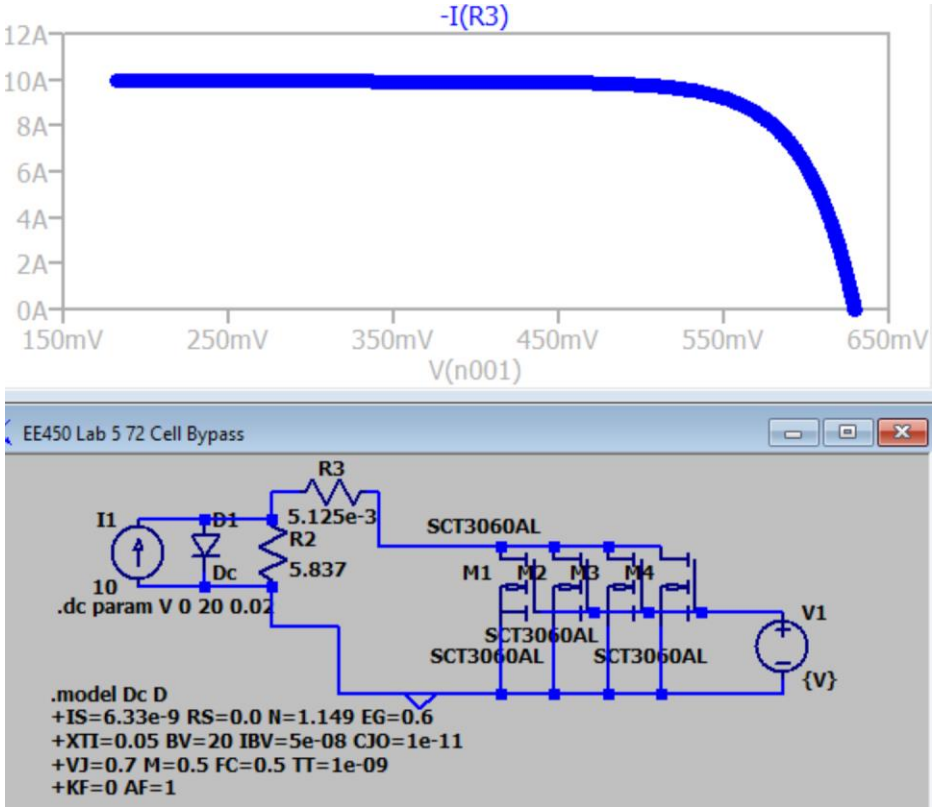


To combine the benefits of both designs, two identical loads featuring four common-gate SCT3060AL mosfets will be placed in parallel, each being controlled independently. For panel tests, the microcontroller will run through 500 load points using the primary load while the secondary load is off (open load). It will then run through another 500 load points while the secondary load is turned on near 3 ohms. The combined data will form a complete IV curve. The process of simulating 8 SCT3060ALs proved too iteratively intense for LT Spice, so a figure could not be provided.

For comparison, a single load is sufficient for a high-resolution curve when testing a single cell, shown in Figure VI. The inability to approach closer to short circuit was due to the resistance of the mosfets themselves, though the data approaches close enough to sufficiently estimate the short circuit characteristics.

FIGURE VI

SINGLE CELL IV CURVE SIMULATION



Load Control

Two MCP4921 digital to analog converters (DACs), one for each load, will be controlled by the MSP432. The DACs have a 12-bit resolution, offering precise conversion of the data output from the microcontroller. The DACs communicate via SPI and thus communicate via only 3 connections, two of which are shared, far fewer than parallel communication DACs. The DACs run off of a 5V supply, and must be amplified to be capable of driving the two electronic loads to short circuit [14].

The SCT3060AL mosfets of the electronic loads are considered fully conducting around +18V gate voltage [11]. To reach this, the DACs will be fed through AD822 FET input op amps, which are capable of up to +30V supplies and rail-to-rail outputs [15]. Coupled with an RKZ-052005 +5V to +20V DC-DC converter supplying the AD822s, the amplified DACs will be able to output a voltage range of 0 to +20V, extending the full operating range of the electronic loads [16]. Resistive networks will establish a 4x amplification for the DAC, such that the max +5V DAC output will produce +20V from the op amps.

Current Monitoring

An INA281A3 current sense amplifier was chosen to measure the current flow of the solar DUT at each loading condition. This current sense amplifier has a common mode input range of -4V to +110V and a supply voltage of -0.3V to +22V, allowing it to take readings within the +90V maximum panel range while also running from the same source as the MSP432 [17].

The A3 variation has a gain of 100V/V. To restrict the output to the maximum +3.3V readable by the MSP432 onboard ADCs, it was decided that +3V would be output when the maximum current of 12A was read [17]. The following equations decided the associated shunt resistor value:

$$\frac{3V}{100V/V} = 0.03V$$
$$\frac{0.03V}{12A} = 0.0025\Omega$$
$$12A * 0.03V = 0.36W$$

A 5W, 2.5mΩ shunt resistor will be paired with the INA281 for reading of currents up to 12A.

Voltage Monitoring

A simple voltage divider will be used for reading the voltage of the solar DUT. To accommodate both the +90V maximum panel requirement and the +3.3V maximum MSP432 ADC input, a 30 to 1 divider will be used. High resistor values of 20 MΩ and 690kΩ were chosen to form the divider to minimize the effect on the IV curve and reduce the power necessary to dissipate.

$$90V * \frac{690k\Omega}{20M\Omega + 690k\Omega} = 3V$$

The divider will be buffered by a simple MCP6002 general use op amp. The supply voltage of +1.8V to +6.0V will allow it to run on the +3.3V rail provided by the microcontroller, and to buffer the entire voltage range supplied by the divider without saturating [18].

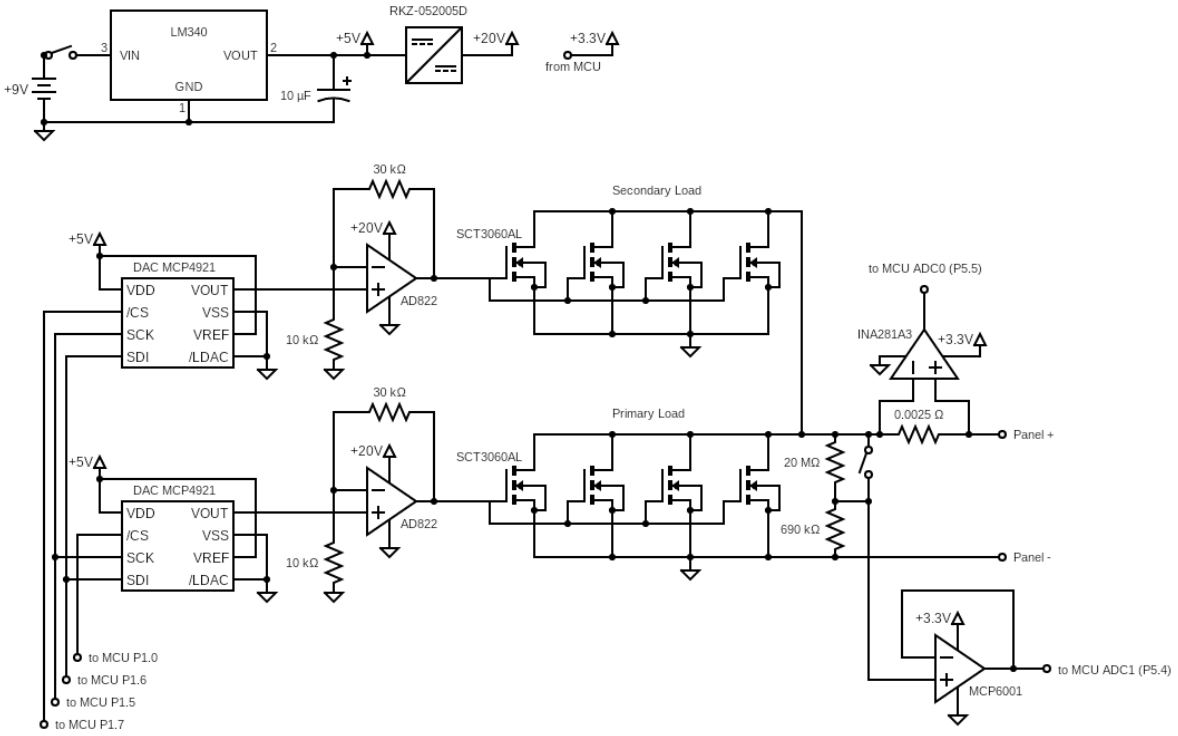
Because the maximum single cell voltage is +0.7V, a voltage divider is unnecessary to safely measure the full voltage range. A physical toggle switch will be implemented to subvert the voltage divider when testing in cell mode.

Circuit Diagram

Figure VII shows the complete interconnection of the solar IV curve tracer circuit.

FIGURE VII

COMPLETE CIRCUIT DIAGRAM



Standalone Modules

Other modules required to meet the outlined project specifications will interface directly with the MSP432 microcontroller and do not require additional circuit design.

A TMP102 will measure ambient temperature, and was chosen for its high resolution and I2C interface. The maximum readable temperature of +85°C surpasses the project specification of +80°C [19].

Irradiance will be measured with a TSL237, which converts light intensity to frequency with a 50% duty cycle square wave. Timer A of the MSP432 will be used to capture this waveform and convert it to an irradiance value [20].

Data will be stored in a 24LC256 256kbyte CMOS EEPROM memory. I2C communication allows it to share a bus with the TMP102, and 256kbytes is more than enough for the planned 10 curve storage [21].

The LCD used will be a generic 1602 16 character 2 line LCD display with parallel communication. The 12 button pad used will also be a generic offering from Adafruit.

Code/User Interface

Code for the MSP432 will be written in Embedded C, based on module header file templates provided by Dr. Amin Malek of the Cal Poly EE Department in his EE-329 Microcontroller-Based System Design course. Upon startup, the LCD display will ask whether the user would like to take a new data set or output an old one. Upon choosing to record a new set, cell or panel modes will be selectable. Each will scale the voltage value differently due to the presence or absence of the voltage divider, and the panel mode will make use of the secondary load. The LCD will ask the user to connect the device to the panel, and after confirming the MSP will run the electronic loads at 1000 test points from short circuit to open load using the DACs and recording the currant and voltage at each using the onboard ADCs. Upon finishing, the user will be given the option to save the data set. If they choose to save, they choose one of ten slots (0-9) to save in, and the device saves the data. The user is then taken back to the initial screen. If the user selects to output old data, the device will ask if they would like to output data from one of the 10 save slots, or output the last data set recorded. The user answers, and the data is output via UART to a personal computer.

Chapter 6.

Construction

Circuit Board

Perforated breadboards were used for this design, as having printed circuit boards manufactured would not be conducive to the constantly changing nature of a prototype, and the cost for having a single board manufactured was rather prohibitive. Parts were placed in proximity to components they interfaced with in accordance to Figure VII. 22 AWG wire was used for interconnections on the main circuit board (see Figure VIII), with solder-bridged pads used to form traces between each pin and associated wire (see Figure IX). The board was planned to be cut to better fit the chassis, but no tools on hand were strong enough.

FIGURE VIII

CIRCUIT BOARD – TOP VIEW

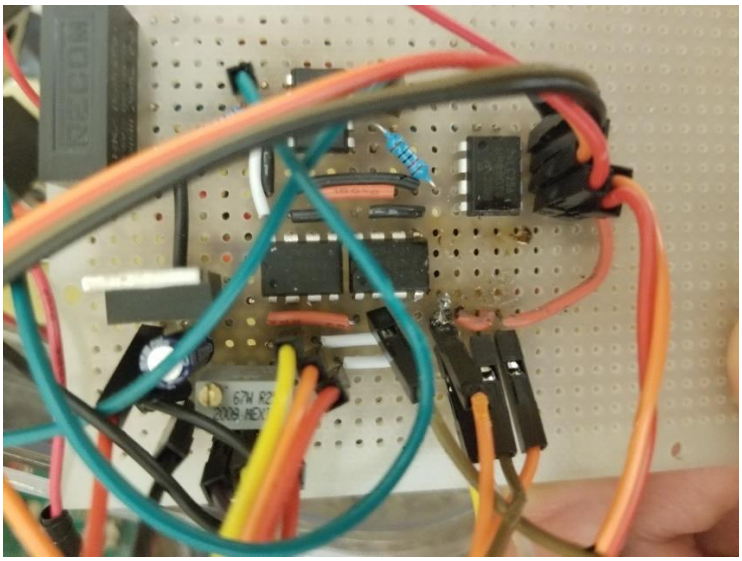
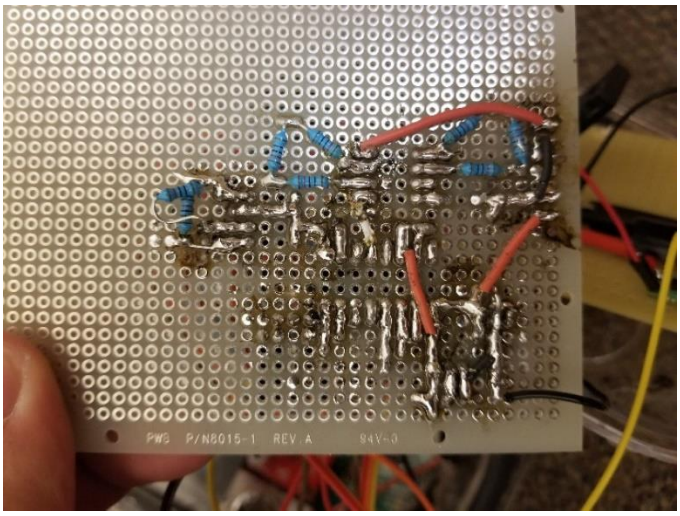


FIGURE IX

CIRCUIT BOARD – BOTTOM VIEW



The electronic load shown in Figure X was connected using 14AWG wire to accommodate the higher current requirements []. The perf board used for the load was unfortunately not plated, so wiring was performed on both sides to hold components in place (see Figure XI). Electrical tape was used to insulate the exposed connections.

FIGURE X

ELECTRONIC LOAD – TOP VIEW

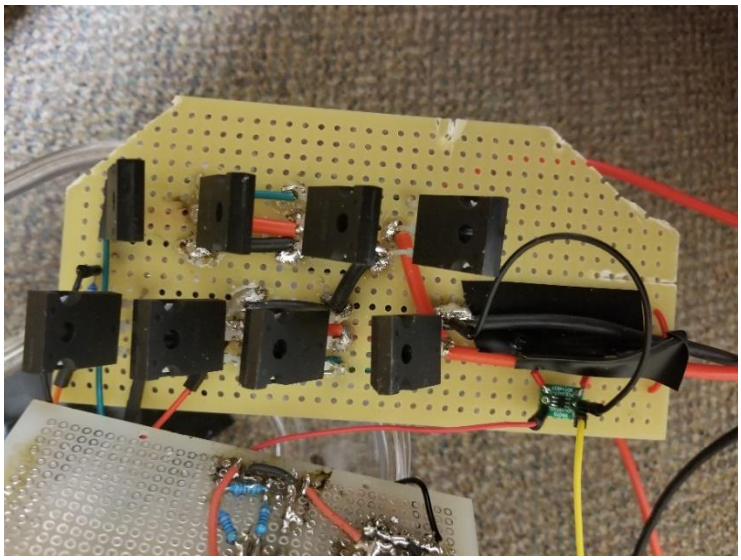
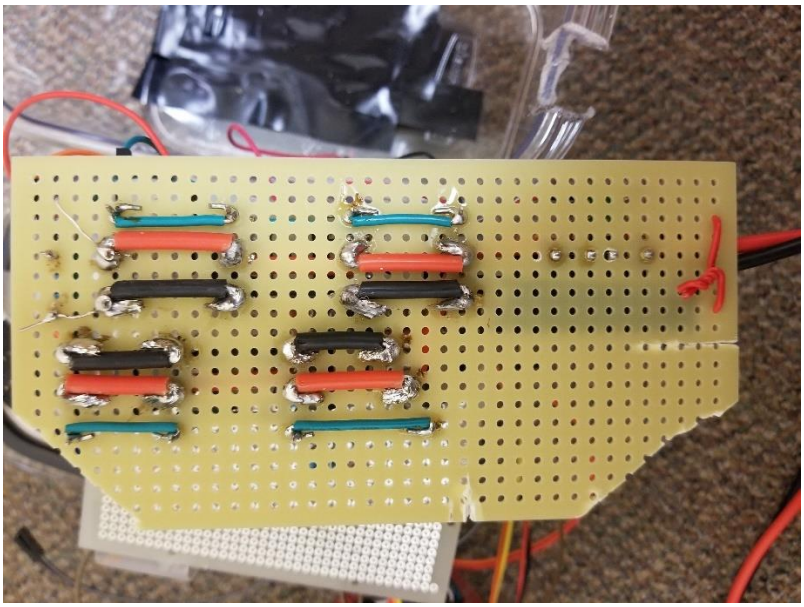


FIGURE XI

ELECTRONIC LOAD – BOTTOM VIEW

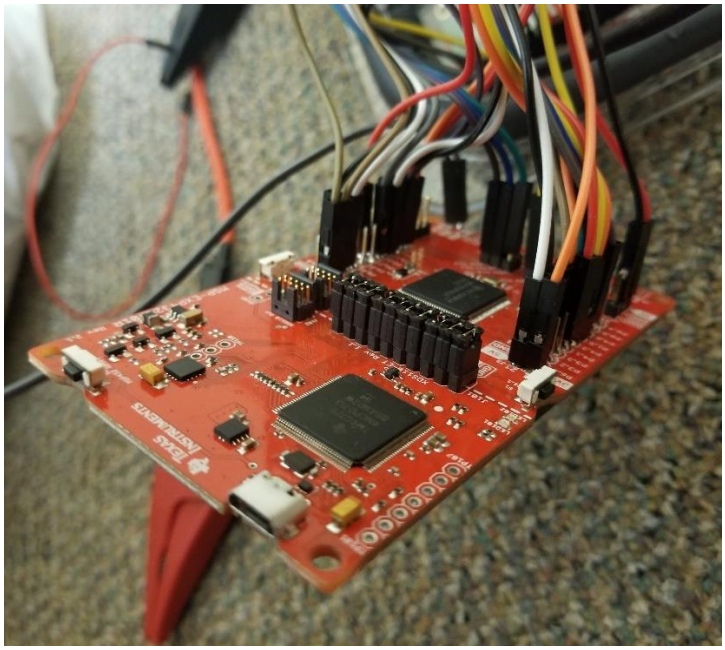


Microcontroller

Generic male to female test leads were used for connecting the MSP432 pins to each IC (see Figure XII). Connections were not soldered to the MSP to allow for future use in other projects.

FIGURE XII

MICROCONTROLLER WIRING



Chassis

The chassis was formed from Tupperware. Holes for wiring and USB access were carved with high heat from a soldering iron, as no other tools were readily available (see Figure XIII). Double sided tape was used to secure the battery pack to the bottom and securely rig the circuit boards within. While primitive, the chassis holds the project together and seals tightly to protect from damage and dirt. 14AWG cables and 20A alligator clamps are used for the solar DUT connection in the final build (see Figure XIV). 6 AA batteries power the project.

FIGURE XIII

CHASSIS LID CONNECTIONS

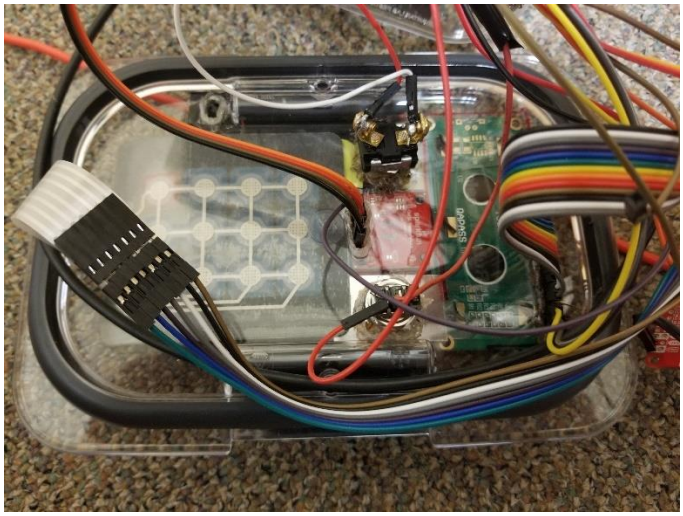
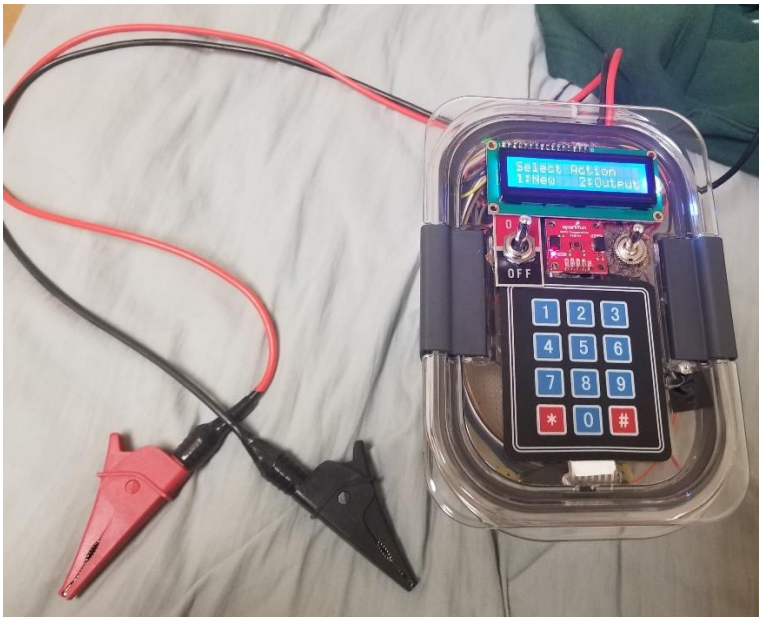


FIGURE XIV

COMPLETED BUILD



Chapter 7.

Testing

Initial Calibration

To calibrate the voltage readings, known voltages were fed to the voltage buffer and read with the corresponding ADC, and changed accordingly to match. The temperature sensor was calibrated with an ice pack. When measured with a multimeter, the electronic load is capable of loading the panel from 20mΩ to 16MΩ, more than enough to test short circuit and open load conditions.

Initial Observations

Tests were performed on a single solar cell (Figure XVI). Variances in voltage and current between two tests were generally due to changes in conditions and not the device itself. After streamlining the code, a single test was confirmed to run in 12 seconds, achieving the 30 second goal. All user commands entered via keypad work, and the LCD correctly displays the current action. Saving to the EEPROM takes around 45 seconds, but recalling for output takes around 10. A serial terminal at a baud rate of 115200 is used for PC communication via USB. Data can successfully be pulled from the serial port (see Figure XV).

FIGURE XV

SINGLE CELL SHORT CIRCUIT CONDITIONS OUTPUT IN .CSV TO TERMINAL

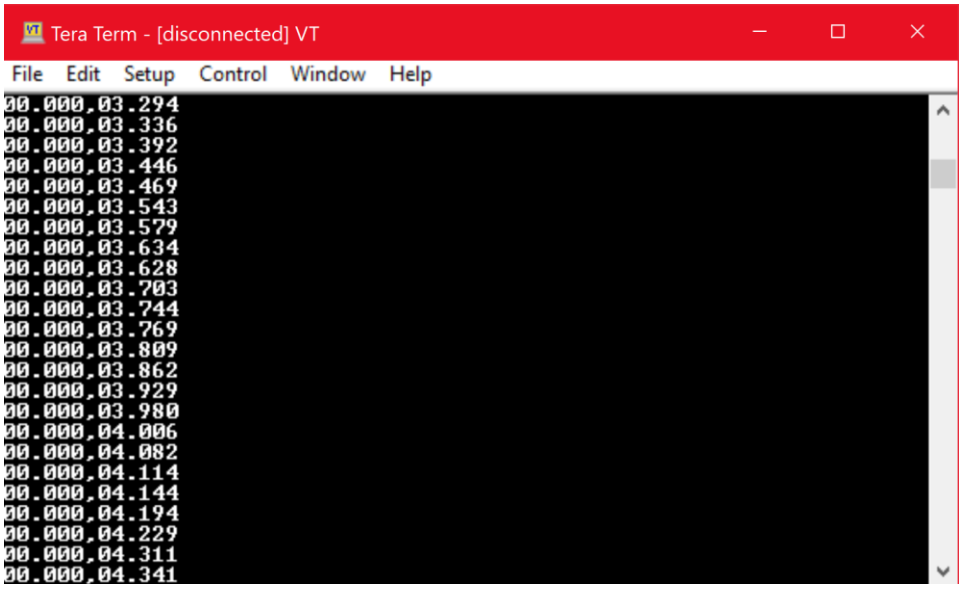
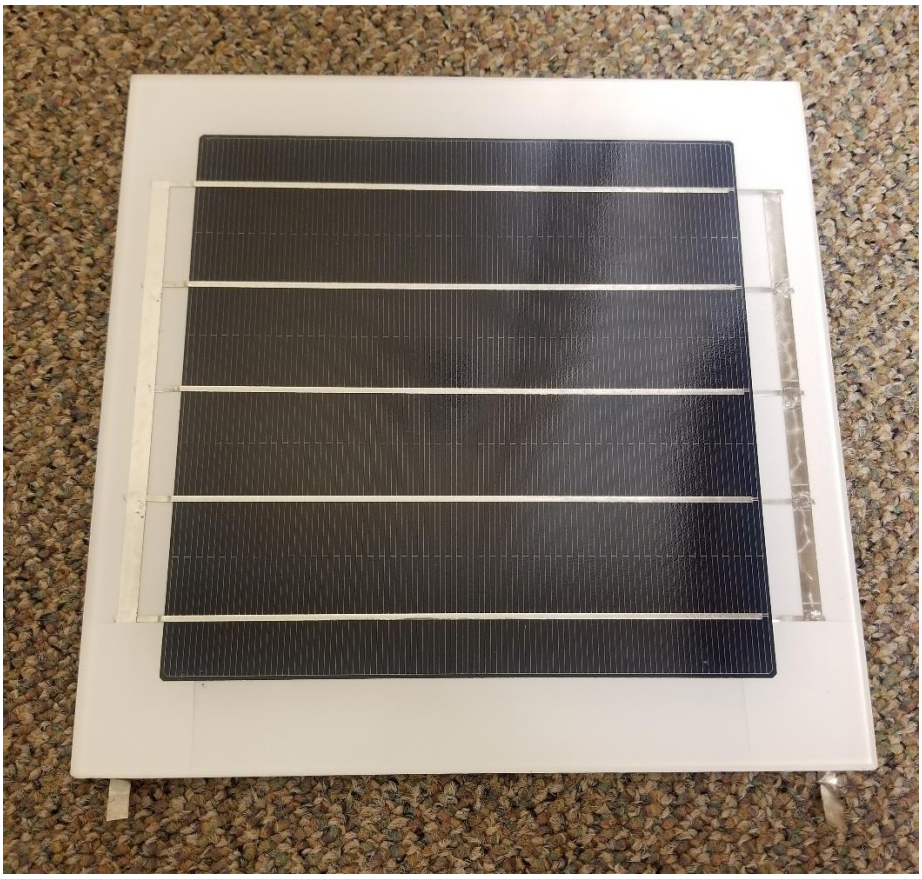


FIGURE XVI

SINGLE CELL DUT

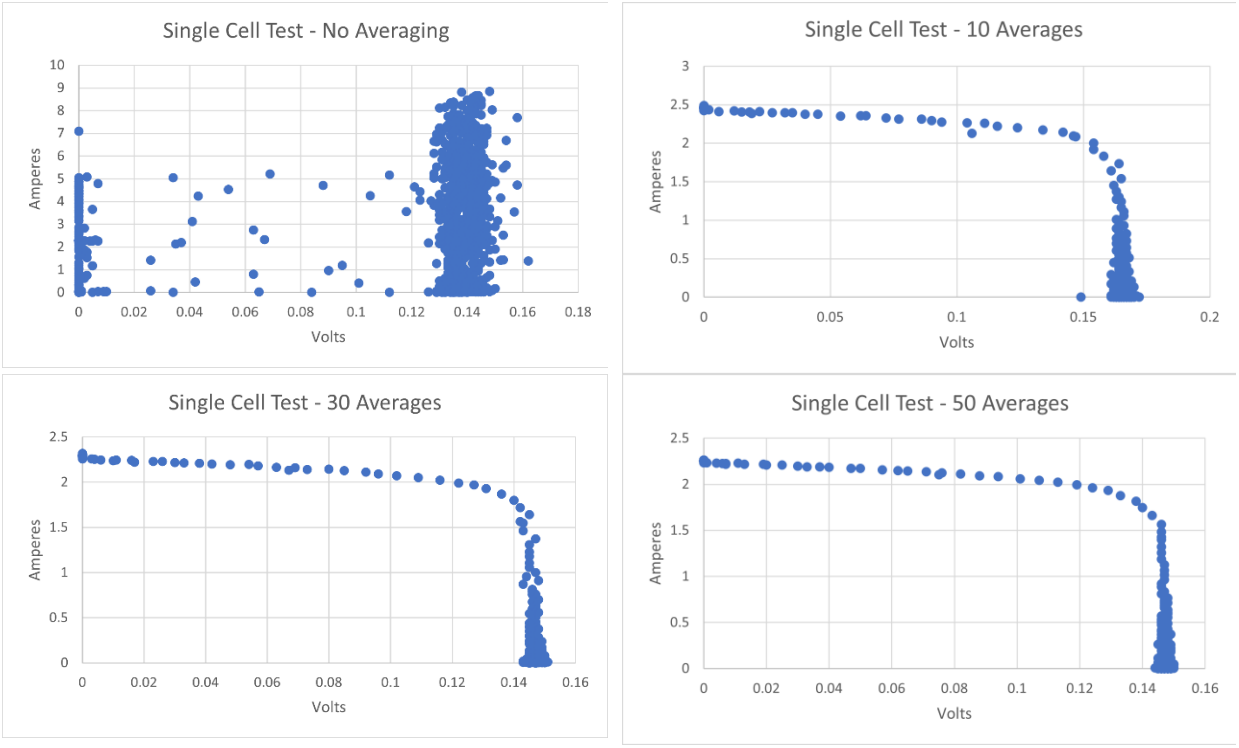


Data Averaging

First runs of the project produced garbage data. It was soon found that the problem was that the collected data was not being averaged. Multiple tests shown in Figure XVII were performed to see the difference between datasets when each point is averaged a different number of times. In the end, 40 averages were chosen due to the high accuracy achieved while keeping the run time under 20 seconds.

FIGURE XVII

IV DATA FOR DIFFERENT AVERAGING AMOUNTS

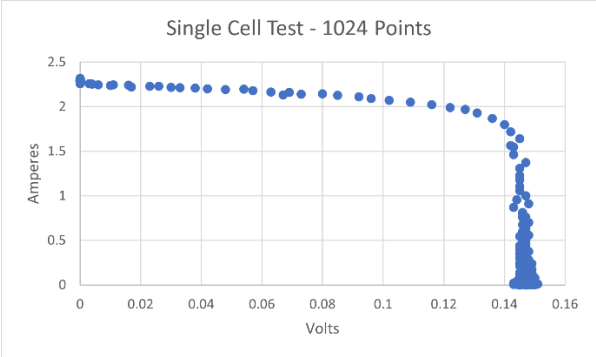


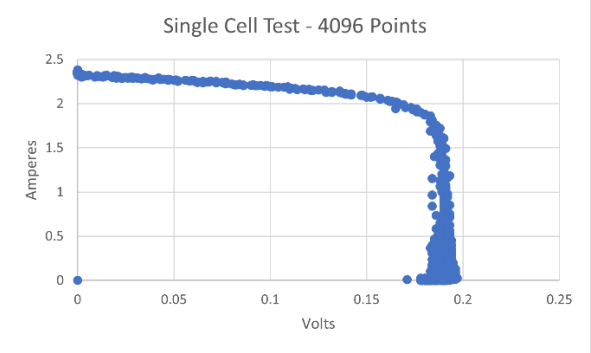
Number of Data Points

As the DACs being used are 12-bit, the maximum number of data points that can be taken in a single run is 4096. Tests were performed using different numbers of load steps, including 1024 and 4096 steps, shown in Figure XVIII. In the end it was decided that while the increased fidelity of the higher number of points is superficially nice, the data being presented is no different or more useful than the 1024 point curve. Prioritizing speed and utility, the 1000-point goal was kept.

FIGURE XVIII

IV DATA FOR DIFFERENT NUMBER OF DATAPOINTS



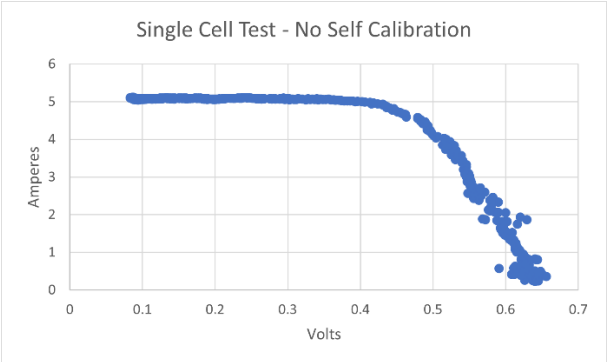


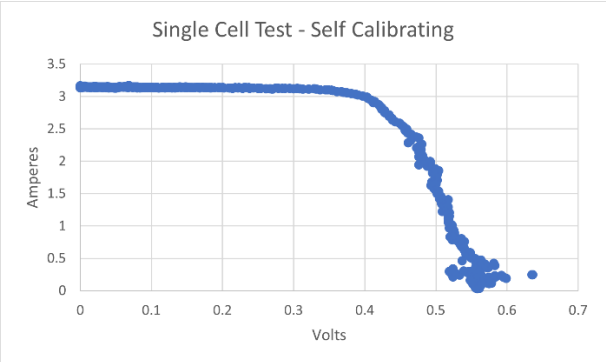
Self-Calibration Each Run

Both voltage and current tended to float an unpredictable amount between tests, making it impossible to subtract a set value from each to obtain correct readings. Instead, code was written to self-calibrate the device's start and end points every run by taking the open circuit current and short circuit voltage (which should both be zero) and subtracting them from every other data point. The difference can be seen in Figure XIX, where the current and voltage initial readings are pulled closer to 0 when auto calibrated.

FIGURE XIX

SELF-CALIBRATING VS NON SELF-CALIBRATING IV CURVES



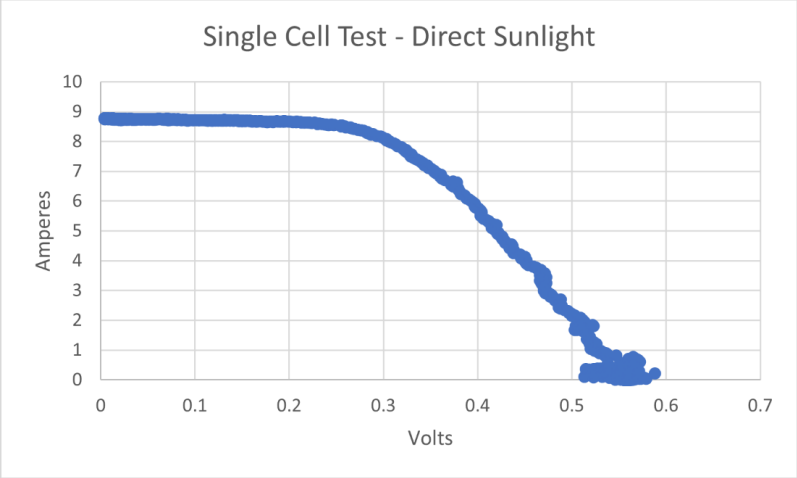


Direct Sunlight – Final Test Run

The final curve logged was from direct sunlight, shown in Figure XX. All previous improvements have been implemented during this test. 18 seconds is the final runtime.

FIGURE XX

FINAL IV CURVE TAKEN

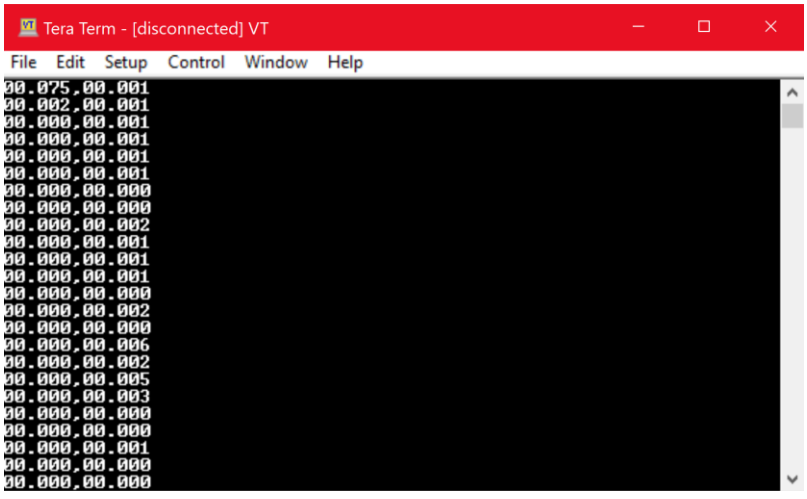


Panel Test – Device Failure

An attempt to log an IV curve of a full 72 cell panel was made. However, after weeks of testing only with a cell, I forgot to flip the witch that runs the voltage divider into the voltage buffer, and upon connecting, the voltage buffer, and eventually microcontroller, shorted (see Figure XXI). The device could not be repaired in time to try again. The cell mode and secondary load were thus not tested during this project.

FIGURE XXI

VOLTAGE READINGS NO LONGER TAKEN



Temperature and Irradiance

Working code was written for the irradiance monitor IC, and data was correctly logged in low light. However, anything brighter than a cell phone light would cause the IC to produce a proportional square wave faster than could be recorded by the microcontroller (>3MHZ). Because of minimal availability of replacements both in person and online at the time, a fix was not made. The irradiance IC was disconnected.

The temperature IC worked well. Temperature was logged in Celsius at the beginning of every test run, with Figure XXII as an example. During the final direct sunlight test, the temperature IC read 25.349 degrees Celsius, compared to the 23 degrees recorded in San Luis Obispo that day.

FIGURE XXII

TEMPERATURE AND IRRADIANCE MEASUREMENTS

Chapter 8.

Conclusions and Future Work

Conclusions

The IV curve tracer design was largely a success. It was able to characterize single cells from short circuit to open load with a full 1024 data points in under 20 seconds, meeting design specifications 2, 5, and 6. Temperature could be successfully measured, but irradiance could not, half fulfilling design specification 1. The electronic load was built to surpass the requirements in specification 3, safely running up to 660W, 100V, and 13.2A with the measurement ICs configured as they are. With protection improvements to the low voltage components, the device will be safer to use on high voltage panels and likely not incur the troubles experience here. The working LCD satisfies specification 4, and successful transfer of data to Excel satisfies specification 7. The device has a plastic casing and runs off of 6 AA batteries, fulfilling 9 and 10. And while the device cannot currently be weighted, the chassis and low part count likely kept it below the 10 pound mark of specification 8.

The project reused a microcontroller, keypad, and LCD required from previous classes, making this a simple and economic project for students in solar classes. The ability to see the IV curve of solar arrays, including the open load, short circuit, and maximum power point values, would be invaluable to understanding their function without spending thousands of dollars for a commercial product.

Future Improvements

Future changes to the tracer design would include a protection diode across the terminals of the voltage sense buffer, to protect the microcontroller from damage if ran in the incorrect mode in the future. Likewise, the code could be changed to short the loads until ready for the test, so even if connected to a panel while in cell mode, the voltage would not exceed safe values until reminded by the device to switch the mode.

Fabrication of a professional PCB and chassis would also improve this project. Covid restrictions on lab access and cost of single board manufacturing limited their use in this project, but future students with better access, and via group purchases in a lab course, would easily be able to acquire a professionally made PCB and sturdier housing.

References

[1] D. Jordan and S. Kurtz, “Photovoltaic Degradation Rates – An Analytical Review,” National Renewable Energy Laboratory, Golden, CO, June 2012. Available: *https://www.nrel.gov/docs/fy12osti/51664.pdf*

[2] Y. Chanchangi, A. Ghosh, S. Sundaram and T. Mallick, “An analytical indoor experimental study on the effect of soiling on PV, focusing on dust properties and PV surface material,” *Solar Energy*, vol. 203, pp 46-68, 2020. Available: *https://www.sciencedirect.com/science/article/pii/S0038092X20303352#*!

[3] L. Hassaine, A. Mraoui and M. Khelif, "Low cost electronic load for out-door testing of photovoltaic panels," 2014 5th International Renewable Energy Congress (IREC), Hammamet, 2014, pp. 1-6, doi: 10.1109/IREC.2014.6826944. Available: *https://ieeexplore.ieee.org/document/6826944* [Accessed Oct 12, 2020].

[4] *I-V500w 1500V 15A I-V Curve Tracer compatible with ht analysis*, HT Instruments, June 3, 2019. Available: *https://www.ht-instruments.us/en-us/products/photovoltaic-testers/i-v-curve-tracers/i-v500w/*

[5] *Solmetric PV Analyzer Kit PVA-1000S 20A,* Web Solar, 2014. [Online]. Available: *https://webosolar.com/product/solmetric-pv-analyzer-kit-pva-1000s/?utm_source=Google%20Shopping&utm_campaign=All%20Products%20Google%20Merchant%20&utm_medium=cpc&utm_term=1559&gclid=Cj0KCQiA48j9BRC-ARIsAMQu3WRSzj1Aicra_VwNcZwy-VcFA7xdUzRtUgBobDBa4INhyNjiMoWsl8oaAt_nEALw_wcB*

[6] *Highest Recorded Temperature on Earth,* Guinness World Records, Death Valley, CA, 1913. [Online]. Available: *https://www.guinnessworldrecords.com/world-records/highest-recorded-temperature*

[7] A Goetzberger and V. U. Hoffmann, *Photovoltaic Solar Energy Generation*. Springer, 2005.

[8] “National Electrical Code,” National Fire Protection Association, Quincy, MA, 2014.

[9] *World’s Largest Selection of Electronics Components,* Digi-Key, Thief River Falls, MN, 2020. [Online]. Available: *https://www.digikey.com/*

[10] Texas Instruments, “MSP432P401R, MSP432P401M SimpleLink™ Mixed-Signal Microcontrollers,” MSP432P401R datasheet, Mar. 2015. Available: *https://www.ti.com/lit/ds/symlink/msp432p401r.pdf?ts=1602553608110&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FMSP432P401R*

[11] ROHM Semiconductor, “N-Channel SiC power MOSFET,” SCT3120AL datasheet, 2015. Available: *https://fscdn.rohm.com/en/products/databook/datasheet/discrete/sic/mosfet/sct3120al-e.pdf*

[12] *Entry Level Salary for Electrical Engineers,* CollegeGrad, 2020. [Online]. Available: *https://collegegrad.com/salaries/electrical-engineer*

[13] Texas Instruments, "LM340 Family Wide Vin 1.5-A Fixed Voltage Regulators," LM340 datasheet, Sept. 2016. Available: *https://www.ti.com/lit/ds/symlink/lm340.pdf?ts=1623594465178&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FLM340%253Futm_source%253Dsupplyframe%2526utm_medium%253DSEP%2526utm_campaign%253Dnot_alldatasheet%2526DCM%253Dyes%2526dclid%253DCjgKEAjw2ZaGBhC1mbjo2Oed62ISJAA-HcSWr0AV2LS8DVuT3c3lr59awc5e_56vDH0LfWcAJPr3pPD_BwE*

[14] Microchip, "12-Bit DAC with SPI Interface," MCP4921 datasheet, 2007. Available: *http://ww1.microchip.com/downloads/en/devicedoc/21897b.pdf*

[15] Analog Devices, "Single-Supply, Rail-to-Rail, Low Power Fet-Input Op Amp," AD822 datasheet, 2015. Available: *https://www.analog.com/media/en/technical-documentation/data-sheets/AD822.pdf*

[16] RECOM, "DC/DC Converter," RKZ-052005D datasheet, 2018. Available: *https://recom-power.com/pdf/Econoline/RKZ-xx2005.pdf*

[17] Texas Instruments, "INA281, -4 to 110-V, 1.3-MHz Current-Sense Amplifier," INA281A3 datasheet, 2020. Available: *https://www.ti.com/lit/ds/symlink/ina281.pdf?ts=1623612818825&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FINA281*

[18] Microchip, "1 MHz, Low-Power Op Amp," MCP6001 datasheet, 2020. Available: *https://ww1.microchip.com/downloads/en/DeviceDoc/MCP6001-1R-1U-2-4-1-MHz-Low-Power-Op-Amp-DS20001733L.pdf*

[19] Texas Instruments, "Low Power Digital Temperature Sensor With SMBus/Two-Wire Serial Interface in SOT563," TMP102 datasheet, 2007. Available: *https://www.sparkfun.com/datasheets/Sensors/Temperature/tmp102.pdf*

[20] AMS, "High-Sensitivity Light-to-Frequency Converter," TSL237 datasheet, 2018. Available: *https://ams.com/documents/20143/36005/TSL237_DS000156_3-00.pdf/4aa35672-5c5e-3bb7-4d6b-92f4c76a3531*

[21] Microchip, "256K I2C Serial EEPROM," 24LC256 datasheet, 2019. Available: *https://ww1.microchip.com/downloads/en/DeviceDoc/24AA256-24LC256-24FC256-Data-Sheet-20001203W.pdf*

[22] "Equipment Authorization – RF Device," Federal Communications Commission, 20-Mar-2018. [Online]. Available: *https://www.fcc.gov/oet/ea/rfdevice*

[23] “DIRECTIVE 2011/65/EU OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 8 June 2011 on the restriction of the use of certain hazardous substances in electrical and electronic equipment,” EUR, 01-Sep-2011. Available: *https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32011L0065*

Appendix A.

ABET Senior Project Analysis

Project Title: Solar IV Curve Tracer

Student's Name: Jonathan Skelly

Student's Signature:

Advisor's Name: Professor Dale Dolan

Advisor's Initials: **Date:**

1. Summary of Functional Requirements

This device is designed to be attached to a solar array's output terminals. By cycling a variable load between 0 and 450W, the device logs solar array current and voltage data at 1000 different operating points, and outputs this data to a computer interface. The device logs irradiance up to 1000 W/m^2, and temperature up to 85 degrees Celsius, so IV data can be correctly interpreted. The device is under 10 pounds, capable of running automatically at a button press, and can test both a single cell (0-0.7V/0-12A) and a single module (0-90V/0-12A). The device will complete operation and log a full IV curve in under 30 seconds.

2. Primary Constraints

- Design Challenges: The largest design challenge is the high input voltage and current provided by the panel. The project must be capable of taking 12A and 600V, limiting the component choices to high voltage tolerances. Resolution must also be high enough to allow for graphing of single-cell IV curves, which would have voltages under 1V (most likely requiring operation at the lower bound 100-point graphing mode, restricting higher numbers of points).

- Project Difficulties: Dealing with very high power, current, and voltage was the largest difficulty of the project. Few components work at 90V and 450W, and those that do get expensive. Proper voltage scaling was needed for each component to ensure compatibility with both the electronic load and the microcontroller. Large-AWG wire was necessary to safely carry the required current load. Failure of the device due to high voltage damage only emphasizes the difficulty of working within such parameters.

3. Economic

- Economic Impacts: The product will have a positive impact on financial capital, as the student-led, condensed design will result in a inexpensive IV tracer than those currently available. The lower production and purchase cost will increase human capital because the price point will make the tracer more readily available to students and engineers, helping increase productivity and knowledge in the solar field. The tracer's ability to give the data necessary to improve solar module design and operation should increase manufactured capital, in terms of electricity produced per panel. Improved solar operation will increase natural capital by reducing the amount of electricity needed from nonrenewable sources.

- Costs & Benefits: Costs associated with the IV curve tracer will be almost entirely up front with the initial purchase, with small costs for batteries throughout its life. Benefits will constantly accrue, as a solar array designed and maintained with an IV tracer will have been optimized for the required load and highest power point, providing higher efficiency for the duration of the array's use.

- Project Inputs: The inputs to this project are the required parts and man-hours needed to complete the design. Components and labor cost at minimum wage totaled, the project initial design is projected to cost $4185. Labor cost will not be paid, and ROHM Semiconductor will be providing SiC MOSFETs.

- Actual Final Cost: The final cost of the project was $224.65, not including shipping and labor (see Appendix B).

- Additional Equipment Costs: Additional equipment include a soldering station and a high-voltage test bench or solar panel.

- Who Benefits: Anyone using solar technology has the ability to benefit from this project if purchased. Owners of solar arrays can adjust their own setups for higher power output, and engineers can more easily characterize panels and arrays in design.

- Product Timing: This product should be expected to last at least 10 years, but is capable of lasting much longer if kept in good condition. Maintenance between purchase and obsolescence should not be necessary other than battery changes.

- Estimated Development Time: 9 months.

- Actual Development Time: 9 months.

4. If manufactured on a commercial basis:

- The device most likely will not be manufactured outside of small-scale for university and student use. However, if it were manufactured, the device's use of commonly available parts would allow

production without sourcing proprietary components. This would keep production costs low and allow plants to freely choose from where to purchase their parts. Government standards like FCC certification on electronic interference and RoHS compliance on materials used would have to be followed and assessed through both manufacturing and the final product [22], [23]. A professional chassis and PCB would be necessary for commercial production.

5. Environmental

- Environmental Impacts: Mining and refining processes for required MOSFETs and other silicon devices produce environmentally unsafe chemicals and waste. Batteries used in the project are unsafe to dispose of. However, after production this project will be used to design more efficient solar arrays and learn about their function, thus increasing the amount of solar energy produced and reducing the need for other sources, which can be environmentally beneficial [23].

- Natural Resources Used: The production of the components in this product will use silicon, copper, tin, lead, and ceramics. Less directly, production will also require necessary energy and land to manufacture these components. PCB production uses various manufactured chemical adhesives and fiberglass, while the enclosure will be petroleum-based plastic.

- Resource Impact: The production of components needed for this project may harmfully impact ecosystems and land around mining sites, waterbeds near manufacturing plants, and air from fossil fuels likely used throughout the entire process. The completed project may improve, in small quantities over time, air quality and global warming by improving existing and prompting development of new solar farms.

- Species Impact: The aforementioned harmful natural impacts of production will affect animals in those areas, while making solar arrays easier to design might increase their number, turning habitats into shaded solar farms.

6. Manufacturability

- Manufacturing Issues: A large challenge with manufacturing will be the increased cost of high-power ICs used in the design. Regardless of how the chassis is constructed during the project, it would have to be constructed from molds if built on a commercial scale, requiring entirely different equipment from the circuit fabrication. Reliability would have to be rigorously tested under various loading conditions and temperatures, with capacitors, heatsinks, and other methods employed to resist quick functional and environmental changes that could damage the product in use.

7. Sustainability

- Device Maintenance Issues: The only planned device maintenance should be battery replacement, but potential fuse blows or damage to external cable connections would also be something that could occur through normal use.
- Sustainable Resources: This device improves the use of sustainable resources by allowing for increased PV production from solar arrays.
- Potential Future Upgrades: An on-board display and the ability to handle higher voltages/more panels.
- Challenges for Upgrading: The size and cost of the device would both likely increase with upgrades, reducing the usability of the device.

8. Ethical

Select points of IEEE's and ABET's ethical frameworks are used for assessing this project's methodologies and standards.

- **IEEE:** 1. to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment:

  Potential manufacturing hazards have been disclosed here. Potential usage hazards, relating to interfacing with high-power solar arrays, would be specified and warned against in product instructions and/or via warnings on the product itself, as well as in the Project Report.

- **IEEE:** 6. to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;

  My experience in power electronics has been noted to the project advisor, though is limited in certain areas specific to this project (DC to DC conversion, variable loads). The project advisor will be notified of knowledge and skill limitations as they arise.

- **ABET:** 4. Engineers shall act in professional matters for each employer or client as faithful agents or trustees, and shall avoid conflicts of interest.

  Perceived conflicts of interest might arise from receiving product funding/components from ROHM and facility use from Cal Poly. To avoid conflicts of interest, all contributing parties have been specified before the project has begun, and the specific contributions will be detailed in the Project Report after project completion.

- **ABET:** 5. Engineers shall build their professional reputation on the merit of their services and shall not compete unfairly with others.

  Underhanded, immoral, or unfair practices could set bad precedent for the industry and create an ethical product. Local laws and national engineering guidelines will be studied and observed during the design of this project.

- **ABET:** 7. Engineers shall continue their professional development throughout their careers and shall provide opportunities for the professional development of those engineers under their supervision.

  The ability to plot IV curves of solar arrays will allow students, engineers, new solar owners, and project contributors to get a better understanding of how solar panels work under various loads, how current, voltage, and power are related in solar arrays, and how temperature and irradiance affect solar production. This knowledge can be applied both to current solar panel technology and those in the future.

9. Health and Safety

- Health and Safety Concerns: Chemicals used in silicon refinement and etching can be hazardous to humans and the environment where they are disposed of. Lead, fiberglass, and checmicals used in the PCB and circuit fabrication prossessess could also be harmful to the manufacturers, customers, and environment. The product would have to adhere to RoHS's banning of certain concentrations of hazardous materials to remain a safe and legal product [23]. Using the product on high-power panels (the tracer can be used on up to 450W systems) could present danger via electrocution to the operator if mishandled. Battery acid via old or damaged batteries present a minor health hazard to the user.

10. Social and Political

- Social and Political Issues: This product is obviously influenced by, and indirectly influences, the political debate about the expansion of solar and green energy. The required components fall under social and political debates about the ethics of mining and manufacturing.

- Stakeholders: Solar panel owners are direct stakeholders, as the use of this project will directly influence their decisions related to solar design and maintenance. Companies who produce the required components are also primary stakeholders, who are paid for the production of the product. Secondary stakeholders include those who receive cleaner air by the hopeful eventual efficiency boost in solar PV production, as well as those living in close proximity to the component production facilities and will experience the harmful effects of waste products.

- Inequities: This project may create inequities between solar farms that can afford and use an IV tracer, and thus have more efficient solar production, versus those that cannot. Inequities between those that must directly deal with the waste and land usage of production facilities, as opposed to those that live elsewhere and do not, is also apparent.

11. Development

- New Techniques: The design and implementation of electronic loads, the handling of high power, and interfacing simultaneously with multiple ICs were among the subjects studied for completion of this project. While such subjects had been touched on before, combining them all into a single comprehensive project had not previously been done in my college career.

- Literature Search: See References.

Appendix B.

Final Parts List and Cost

The full list of parts and their respective costs are listed in Table V. Shipping costs, tax, etc. were not factored into the cost because they are highly variable and do not necessarily reflect the cost to complete the project if it were to be replicated. The LCD, Numpad, and MSP were specifically chosen to match those used in required Cal Poly EE/CPE coursework soi students would already have access to them, and thus are not counted in the final cost either.

TABLE V

FINAL COSTS

ITEM	PART NUMBER	QUANTITY	UNIT PRICE	EXTENDED PRICE
SiC Mosfet	SCT3060AL	8	$11.26	$90.08
DAC	MCP4921	2	$2.17	$4.34
DAC Amplifier	AD822	1	$11.33	$11.33
Current Sense Amplifier	INA281A3	1	$2.95	$2.95
2.5 mohm Shunt Resistor		1	$12.31	$12.31
Voltage Divider Buffer	MCP6002	1	$0.36	$0.36
690k Resistor		1	$0.58	$0.58
20M Resistor		1	$0.50	$0.50
Linear Voltage Regulator	LM340	1	$3.26	$3.26
DC-DC Converter	RKZ-052005D	1	$6.26	$6.26
EEPROM	24LC256	1	$0.90	$0.90
Temperature Sensor	TMP102	1	$4.95	$4.95
Irradiance Sensor	TSL237	1	$3.86	$3.86
Micrcocontroller	MSP432P401R	1	Students should already own.	
16x2 LCD Display	CFAH1602	1	Students should already own.	
12 Button Numpad		1	Students should already own.	
14AWG Wire		2	$16.20	$32.40
22AWG Wire		2	$2.00	$4.00
Aligator Clips		1	$2.55	$2.55
1.07mm Hole Perfboard	8015-1	1	$7.73	$7.73
1.70mm Hole Perfboard	17T36WE	1	$10.54	$10.54
Tupperware	4.7 Cup	1	$9.99	$9.99
Toggle Switches		2	$3.50	$7.00
Battery Pack	SBH361A	1	$3.76	$3.76
Miscellaneous		1	-	$5.00
			TOTAL:	$224.65

Appendix C.

MSP432P401R Connections

Table VI describes each pin connection and function in the overall project in relation to the MSP microcontroller. Refer to cited data sheets for specific purpose of each module connection.

TABLE VI

MSP432 PIN CONNECTIONS

MSP432P401R Launchpad Pin Number	Connection
1.0	DAC1 CS
1.5	DAC SCK
1.6	DAC SDI
1.7	DAC2 CS
2.0	Keypad 0
2.1	Keypad 1
2.2	Keypad 2
2.3	Keypad 3
2.4	Keypad 4
2.5	Keypad 5
2.6	Keypad 6
3.5	Temp Alert
3.6	SDA
3.7	SCL
4.0	LCD D0
4.1	LCD D1
4.2	LCD D2
4.3	LCD D3
4.4	LCD D4
4.5	LCD D5
4.6	LCD D6
4.7	LCD D7
5.0	LCD RS
5.1	LCD RW
5.2	LCD EN
5.4	ADC1
5.5	ADC0

Appendix D.

Code (Embedded C)

Main

```
#include "msp.h"
#include <stdint.h>
#include "delayms.h"
#include "SPI.h"
#include "EEPROM.h"
#include "LCD.h"
#include "Keypad.h"
#include "DAC1.h"
#include "DAC2.h"


int readCurrent=0;          // Current Memory read value
int readVoltage=0;          // Voltage Memory read value
int ivcurve[1025][2];    // Parsed voltage value
int cnt;                    // Counter
uint8_t key;
int init = 0;
int pick = 1;
int data = 2;
int save = 3;
int out = 4;
int mode = 1;
int state = 0;
float calibratecur;
float calibratevol;
float avvol;
float avcur;
int volload;
int curload;
int datacnt;
float currentoffset;
float voltageoffset;
uint16_t dacval = 0;
uint16_t dacval2 = 0;
int parsevol[6];
int parsecur[6];
uint8_t HiValue;
uint8_t LoValue;
uint16_t address;
uint32_t i;
int cval;
int avtemp;

#define TEMP_ADDRESS 0x48
```

```
#define EEPROM_ADDRESS 0x50
uint16_t value;


int main(void) {

    // Stop WDT
    WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;

    //LCD setup
    LCD_init();     // Initialize LCD
    clear_LCD();    // Clear LCD
    delay_ms(500, 3000000);


    //DAC setup
    SPI_init();     // Initialize SPI pins


    //Keypad Setup
    keypad_init();              // setup gpio pins for keypad

    //ADC Setup
    P5->SEL1 |= BIT4;           // Configure P5.4 for voltage ADC
    P5->SEL0 |= BIT4;
    P5->SEL1 |= BIT5;           // Configure P5.5 for current ADC
    P5->SEL0 |= BIT5;
    ADC14->CTL0 = ADC14_CTL0_SHT0_7 | ADC14_CTL0_SHP | ADC14_CTL0_ON | ADC14_CTL0_CONSEQ_1;
    ADC14->CTL1 = ADC14_CTL1_RES_3;     // Use sampling timer, 14-bit conversion
    ADC14->MCTL[1] = ADC14_MCTLN_INCH_1 | ADC14_MCTLN_EOS; // A1 ADC input select; End of
Sequence
    ADC14->MCTL[0] |= ADC14_MCTLN_INCH_0; // A0 ADC input select
    ADC14->CTL0 |= ADC14_CTL0_ENC;        // Enable conversions

    //UART Setup
    P1->SEL0 |= 0x0C;           // P1.3, P1.2 for UART
    P1->SEL1 &= ~0x0C;
    EUSCI_A0->CTLW0 |= 1;       /* put in reset mode for config */
    EUSCI_A0->MCTLW = 0;        /* disable oversampling */
    EUSCI_A0->CTLW0 = 0x0081;   /* 1 stop bit, no parity, SMCLK, 8-bit data */
    EUSCI_A0->BRW = 26;         /* 3,000,000 / 115200 = 26 */
    EUSCI_A0->CTLW0 &= ~1;      /* take UART out of reset mode */

    // Enable global interrupt
    __enable_irq();

    //main
    while (1){

        if (state == init){
```

```
while(key!=0xFF){
    key = keypad_getkey();  // read the keyboard value
}

//User picks function
first_line_LCD();
write_char_LCD('S');
write_char_LCD('e');
write_char_LCD('l');
write_char_LCD('e');
write_char_LCD('c');
write_char_LCD('t');
write_char_LCD(' ');
write_char_LCD('A');
write_char_LCD('c');
write_char_LCD('t');
write_char_LCD('i');
write_char_LCD('o');
write_char_LCD('n');
second_line_LCD();
write_char_LCD('1');
write_char_LCD(':');
write_char_LCD('N');
write_char_LCD('e');
write_char_LCD('w');
write_char_LCD(' ');
write_char_LCD(' ');
write_char_LCD(' ');
write_char_LCD('2');
write_char_LCD(':');
write_char_LCD('O');
write_char_LCD('u');
write_char_LCD('t');
write_char_LCD('p');
write_char_LCD('u');
write_char_LCD('t');

key=0;
while(key==0){
    key = keypad_getkey();  // read the keyboard value
    if(key!=1 & key!=2){    // ignore extraneous selections
        key=0;
    }
}
if(key==1){
    state=pick;     //Cell mode
}
else{
    state=out;     //Panel mode
}
clear_LCD();    // Clear LCD
```

```
    first_line_LCD();
}

if (state == pick){

    while(key!=0xFF){
        key = keypad_getkey();  // read the keyboard value
    }

    //User picks data mode
    first_line_LCD();
    write_char_LCD('S');
    write_char_LCD('e');
    write_char_LCD('l');
    write_char_LCD('e');
    write_char_LCD('c');
    write_char_LCD('t');
    write_char_LCD(' ');
    write_char_LCD('M');
    write_char_LCD('o');
    write_char_LCD('d');
    write_char_LCD('e');
    second_line_LCD();
    write_char_LCD('1');
    write_char_LCD(':');
    write_char_LCD('C');
    write_char_LCD('e');
    write_char_LCD('l');
    write_char_LCD('l');
    write_char_LCD(' ');
    write_char_LCD(' ');
    write_char_LCD(' ');
    write_char_LCD('2');
    write_char_LCD(':');
    write_char_LCD('P');
    write_char_LCD('a');
    write_char_LCD('n');
    write_char_LCD('e');
    write_char_LCD('l');

    //User inputs mode choice
    key=0;
    while(key==0){
        key = keypad_getkey();  // read the keyboard value
        if(key!=1 & key!=2){    // ignore extraneous selections
            key=0;
        }
    }
    if(key==1){
        mode=1;          //Cell mode
```

```
    }
    else{
        mode=30.98;     //Panel mode
    }
    clear_LCD();        // Clear LCD
    state=data;
}


else if (state == data){

    while(key!=0xFF){
        key = keypad_getkey();  // read the keyboard value
    }

    //Tells user to connect data probes
    first_line_LCD();
    write_char_LCD('C');
    write_char_LCD('o');
    write_char_LCD('n');
    write_char_LCD('n');
    write_char_LCD('e');
    write_char_LCD('c');
    write_char_LCD('t');
    write_char_LCD(' ');
    write_char_LCD('C');
    write_char_LCD('a');
    write_char_LCD('b');
    write_char_LCD('l');
    write_char_LCD('e');
    write_char_LCD('s');
    write_char_LCD('.');
    second_line_LCD();
    write_char_LCD('R');
    write_char_LCD('e');
    write_char_LCD('a');
    write_char_LCD('d');
    write_char_LCD('y');
    write_char_LCD('?');
    write_char_LCD(' ');
    write_char_LCD(' ');
    write_char_LCD('1');
    write_char_LCD(':');
    write_char_LCD('Y');
    write_char_LCD('e');
    write_char_LCD('s');


    //User inputs confirmation
    key=0;
    while(key==0){
```

```
    key = keypad_getkey();  // read the keyboard value
    if(key!=1){    // ignore extraneous selections
        key=0;
    }
}

clear_LCD();    // Clear LCD
first_line_LCD();
write_char_LCD('W');
write_char_LCD('a');
write_char_LCD('i');
write_char_LCD('t');
write_char_LCD('.');
write_char_LCD('.');
write_char_LCD('.');


//Read temperature
InitI2C(TEMP_ADDRESS);      // Initialize I2C
WriteTemp(0x01, 0x15C);      // Write configuration to Temp sensor
while(cnt<5){
    if(P3->IN & BIT5){          // If data ready flag high,
        value = ReadTemp(0x00); // Read value from Temp reg
        cval = (value)*0.0078125*1000;    // Calculate Celsius * 10
        avtemp+=cval;
        cnt+=1;
    }
}
avtemp/=5;
ivcurve[0][0]=avtemp;


//Set step value for DAC
datacnt = 1;
dacval = 0;
dacval2= 0;
DAC2(dacval2);

// Calculate Voltage Offset
DAC1(4095);
for(cnt=0; cnt<100; cnt++){

    ADC14->CTL0 |= ADC14_CTL0_SC;  // Sample current and voltage
    while(ADC14->IFGR0 != 1){}     // Wait until conversions complete
    ADC14->CTL0 |= ADC14_CTL0_SC;
    while(ADC14->IFGR0 != 3){}     // Wait until conversions complete
    readVoltage=ADC14->MEM[1];     // Save memory value in a variable
    readCurrent=ADC14->MEM[0];     // Save memory value in a variable
    calibratevol=(mode*0.0002*readVoltage)*1000;
    avvol+=calibratevol;
}
```

```
avvol/=100;
voltageoffset=avvol;

// Calculate Current Offset
DAC1(0);
for(cnt=0; cnt<100; cnt++){
    ADC14->CTL0 |= ADC14_CTL0_SC;  // Sample current and voltage
    while(ADC14->IFGR0 != 1){}     // Wait until conversions complete
    ADC14->CTL0 |= ADC14_CTL0_SC;
    while(ADC14->IFGR0 != 3){}     // Wait until conversions complete
    readVoltage=ADC14->MEM[1];     // Save memory value in a variable
    readCurrent=ADC14->MEM[0];     // Save memory value in a variable
    calibratecur=(0.0002*readCurrent)*1000;
    avcur+=calibratecur;
}
avcur/=100;
currentoffset=avcur;


 // Sample until DAC reaches max
 while (dacval < 4096){
     DAC1(dacval);

     for(cnt=0; cnt<40; cnt++){

     ADC14->CTL0 |= ADC14_CTL0_SC;  // Sample current and voltage
     while(ADC14->IFGR0 != 1){}     // Wait until conversions complete
     ADC14->CTL0 |= ADC14_CTL0_SC;
     while(ADC14->IFGR0 != 3){}     // Wait until conversions complete

     readVoltage=ADC14->MEM[1];     // Save memory value in a variable
     readCurrent=ADC14->MEM[0];     // Save memory value in a variable

     calibratecur=(1000*4*0.0002*readCurrent)-currentoffset;   // Calibrate readings
to 1mV value
     calibratevol=(mode*1000*0.0002*readVoltage)-voltageoffset;

     avvol+=calibratevol;
     avcur+=calibratecur;
     }
     avvol/=40;
     avcur/=40;

     if (avcur<0){
         avcur=0;
     }
     if (avvol<0){
         avvol=0;
     }

     ivcurve[datacnt][0]=volload;            // Store in array
```

```
    ivcurve[datacnt][1]=curload;
    datacnt+=1;                    // Increment array address
    dacval+=4;

}

// User chooses to save data
while(key!=0xFF){
    key = keypad_getkey();  // read the keyboard value
}
clear_LCD();    // Clear LCD
first_line_LCD();
write_char_LCD('C');
write_char_LCD('o');
write_char_LCD('m');
write_char_LCD('p');
write_char_LCD('l');
write_char_LCD('e');
write_char_LCD('t');
write_char_LCD('e');
write_char_LCD('.');
write_char_LCD(' ');
write_char_LCD('S');
write_char_LCD('a');
write_char_LCD('v');
write_char_LCD('e');
write_char_LCD('?');
second_line_LCD();
write_char_LCD('1');
write_char_LCD(':');
write_char_LCD('Y');
write_char_LCD('e');
write_char_LCD('s');
write_char_LCD(' ');
write_char_LCD(' ');
write_char_LCD(' ');
write_char_LCD('2');
write_char_LCD(':');
write_char_LCD('N');
write_char_LCD('o');

//User inputs choice
key=0;
while(key==0){
    key = keypad_getkey();  // read the keyboard value
    if(key!=1 & key!=2){    // ignore extraneous selections
        key=0;
    }
}

if(key==1){
```

```
        state=save;
    }
    else{
        state=init;
    }

    clear_LCD();    // Clear LCD
    delay_ms(500, 3000000);

}


else if(state==save){

    InitI2C(EEPROM_ADDRESS);      // Initialize I2C
    while(key!=0xFF){
        key = keypad_getkey();  // read the keyboard value
    }

    //User chooses save slot
    clear_LCD();    // Clear LCD
    first_line_LCD();
    write_char_LCD('C');
    write_char_LCD('h');
    write_char_LCD('o');
    write_char_LCD('o');
    write_char_LCD('s');
    write_char_LCD('e');
    write_char_LCD(' ');
    write_char_LCD('s');
    write_char_LCD('a');
    write_char_LCD('v');
    write_char_LCD('e');
    write_char_LCD(' ');
    write_char_LCD('s');
    write_char_LCD('l');
    write_char_LCD('o');
    write_char_LCD('t');
    second_line_LCD();
    write_char_LCD('(');
    write_char_LCD('0');
    write_char_LCD('-');
    write_char_LCD('9');
    write_char_LCD(')');

    key=0xFF;
    while(key==0xFF){
        key = keypad_getkey();  // read the keyboard value
        if(key>9){              // ignore extraneous selections
            key=0xFF;
        }
```

```
    }

    clear_LCD();    // Clear LCD
    first_line_LCD();
    write_char_LCD('S');
    write_char_LCD('a');
    write_char_LCD('v');
    write_char_LCD('i');
    write_char_LCD('n');
    write_char_LCD('g');
    write_char_LCD('.');
    write_char_LCD('.');
    write_char_LCD('.');

    address=4100*key;

     //Store data
    for(cnt=0; cnt<1025; cnt++){

        address+=1;
        HiValue = ivcurve[cnt][0] >> 8;     //Split data into bytes
        LoValue = ivcurve[cnt][0] & 0xFF;
        WriteEEPROM(address, HiValue);      //Write bytes starting at given address
        for (i = 4000; i > 0; i--); // Delay for EEPROM write cycle (5 ms)

        address+=1;
        WriteEEPROM(address, LoValue);
        for (i = 4000; i > 0; i--); // Delay for EEPROM write cycle (5 ms)

        address+=1;
        HiValue = ivcurve[cnt][1] >> 8;
        LoValue = ivcurve[cnt][1] & 0xFF;
        WriteEEPROM(address, HiValue);
        for (i = 4000; i > 0; i--); // Delay for EEPROM write cycle (5 ms)

        address+=1;
        WriteEEPROM(address, LoValue);
        for (i = 4000; i > 0; i--); // Delay for EEPROM write cycle (5 ms)
    }


    state=init;

}

else if(state==out){
    while(key!=0xFF){
        key = keypad_getkey();  // read the keyboard value
    }
    //User chooses to output last curve taken, or from memory
    clear_LCD();    // Clear LCD
```

```
first_line_LCD();
write_char_LCD('U');
write_char_LCD('s');
write_char_LCD('e');
write_char_LCD(' ');
write_char_LCD('l');
write_char_LCD('a');
write_char_LCD('s');
write_char_LCD('t');
write_char_LCD(' ');
write_char_LCD('o');
write_char_LCD('r');
write_char_LCD(' ');
write_char_LCD('l');
write_char_LCD('o');
write_char_LCD('a');
write_char_LCD('d');
second_line_LCD();

write_char_LCD('1');
write_char_LCD(':');
write_char_LCD('L');
write_char_LCD('o');
write_char_LCD('a');
write_char_LCD('d');
write_char_LCD(' ');
write_char_LCD(' ');
write_char_LCD(' ');
write_char_LCD('2');
write_char_LCD(':');
write_char_LCD('L');
write_char_LCD('a');
write_char_LCD('s');
write_char_LCD('t');

key=0;
while(key==0){
    key = keypad_getkey();  // read the keyboard value
    if(key!=1 & key!=2){    // ignore extraneous selections
        key=0;
    }
}

if(key==1){

    InitI2C(EEPROM_ADDRESS);      // Initialize I2C

    while(key!=0xFF){
        key = keypad_getkey();  // read the keyboard value
    }
```

```
//User picks which save slot to output
clear_LCD();    // Clear LCD
first_line_LCD();
write_char_LCD('O');
write_char_LCD('u');
write_char_LCD('t');
write_char_LCD('p');
write_char_LCD('u');
write_char_LCD('t');
write_char_LCD(' ');
write_char_LCD('w');
write_char_LCD('h');
write_char_LCD('i');
write_char_LCD('c');
write_char_LCD('h');
second_line_LCD();
write_char_LCD('s');
write_char_LCD('a');
write_char_LCD('v');
write_char_LCD('e');
write_char_LCD(' ');
write_char_LCD('s');
write_char_LCD('l');
write_char_LCD('o');
write_char_LCD('t');
write_char_LCD('(');
write_char_LCD('0');
write_char_LCD('-');
write_char_LCD('9');
write_char_LCD(')');
key=0xFF;
while(key==0xFF){
    key = keypad_getkey();  // read the keyboard value
    if(key>9){    // ignore extraneous selections
        key=0xFF;
    }
}
clear_LCD();    // Clear LCD
first_line_LCD();
write_char_LCD('W');
write_char_LCD('a');
write_char_LCD('i');
write_char_LCD('t');
write_char_LCD('.');
write_char_LCD('.');
write_char_LCD('.');

address=4100*key;

for(cnt=0; cnt<1025; cnt++){
    address+=1;
```

```
        HiValue=(ReadEEPROM(address))<<8;
        address+=1;
        LoValue=ReadEEPROM(address);
        ivcurve[cnt][0]=HiValue+LoValue;
        address+=1;
        HiValue=(ReadEEPROM(address))<<8;
        address+=1;
        LoValue=ReadEEPROM(address);
        ivcurve[cnt][1]=HiValue+LoValue;

    }
}


while(key!=0xFF){
    key = keypad_getkey();  // read the keyboard value
}
clear_LCD();    // Clear LCD
first_line_LCD();
write_char_LCD('C');
write_char_LCD('o');
write_char_LCD('n');
write_char_LCD('n');
write_char_LCD('e');
write_char_LCD('c');
write_char_LCD('t');
write_char_LCD(' ');
write_char_LCD('U');
write_char_LCD('S');
write_char_LCD('B');
write_char_LCD('.');
second_line_LCD();
write_char_LCD('R');
write_char_LCD('e');
write_char_LCD('a');
write_char_LCD('d');
write_char_LCD('y');
write_char_LCD('?');
write_char_LCD(' ');
write_char_LCD(' ');
write_char_LCD('1');
write_char_LCD(':');
write_char_LCD('Y');
write_char_LCD('e');
write_char_LCD('s');


//User inputs confirmation
key=0;
while(key==0){
    key = keypad_getkey();  // read the keyboard value
```

```
    if(key!=1){             // ignore extraneous selections
        key=0;
    }
}

clear_LCD();    // Clear LCD
first_line_LCD();
write_char_LCD('W');
write_char_LCD('a');
write_char_LCD('i');
write_char_LCD('t');
write_char_LCD('.');
write_char_LCD('.');
write_char_LCD('.');



datacnt=0;
while(datacnt<1025){

    //Data is parsed to integers for UART output
    parsevol[0]= (int)((ivcurve[datacnt][0]/10000)+48);   // Parse voltage to array
    parsevol[1]= (int)(((ivcurve[datacnt][0]/1000)%10)+48);
    parsevol[2]= 46;
    parsevol[3]= (int)(((ivcurve[datacnt][0]/100)%10)+48);
    parsevol[4]= (int)(((ivcurve[datacnt][0]/10)%10)+48);
    parsevol[5]= (int)((ivcurve[datacnt][0]%10)+48);

    parsecur[0]= (int)((ivcurve[datacnt][1]/10000)+48);   // Parse current to array
    parsecur[1]= (int)(((ivcurve[datacnt][1]/1000)%10)+48);
    parsecur[2]= 46;
    parsecur[3]= (int)(((ivcurve[datacnt][1]/100)%10)+48);
    parsecur[4]= (int)(((ivcurve[datacnt][1]/10)%10)+48);
    parsecur[5]= (int)((ivcurve[datacnt][1]%10)+48);

    for(cnt=0; cnt<=5; cnt++){
        while(!(EUSCI_A0->IFG & 0x02)) { }
        EUSCI_A0->TXBUF =parsevol[cnt];     // Output each voltage decimal place
    }

    while(!(EUSCI_A0->IFG & 0x02)) { }      // Comma
    EUSCI_A0->TXBUF = 44;


    for(cnt=0; cnt<=5; cnt++){
        while(!(EUSCI_A0->IFG & 0x02)) { }
        EUSCI_A0->TXBUF =parsecur[cnt];     // Output each current decimal place
    }

    for(cnt=0; cnt<=12; cnt++){
```

```
                while(!(EUSCI_A0->IFG & 0x02)) { }  // Backspaces
                EUSCI_A0->TXBUF = 8;
            }

            while(!(EUSCI_A0->IFG & 0x02)) { }      // Line Return
            EUSCI_A0->TXBUF = 10;

            datacnt+=1;                             // Increment array address
        }

        clear_LCD();    // Clear LCD
        first_line_LCD();
        write_char_LCD('C');
        write_char_LCD('o');
        write_char_LCD('m');
        write_char_LCD('p');
        write_char_LCD('l');
        write_char_LCD('e');
        write_char_LCD('t');
        write_char_LCD('e');
        write_char_LCD('.');

        delay_ms(2000,3000000);           // Delay
        state=init;
    }
  }
}
```

SPI.h

```
#define DAC_CS1  BIT0
#define DAC_CS2  BIT7

void SPI_init(void){
    P1->SEL0 = BIT5 | BIT6;     // Set P1.5 and P1.6 as
                                // SPI pins functionality

    P1->DIR |= DAC_CS1;         // set as output for CS
    P1->DIR |= DAC_CS2;         // set as output for CS


    EUSCI_B0->CTLW0 |= EUSCI_B_CTLW0_SWRST; // Put eUSCI state machine in reset

    EUSCI_B0->CTLW0 = EUSCI_B_CTLW0_SWRST    |  // keep eUSCI in reset
                      EUSCI_B_CTLW0_MST      |  // Set as SPI master
                      EUSCI_B_CTLW0_SYNC     |  // Set as synchronous mode
                      EUSCI_B_CTLW0_CKPL     |  // Set clock polarity high
                      EUSCI_B_CTLW0_UCSSEL_2 |  // SMCLK
                      EUSCI_B_CTLW0_MSB;        // MSB first

    EUSCI_B0->BRW = 0x01;              // div by 2 fBitClock = fBRCLK / UCBRx

    EUSCI_B0->CTLW0 &= ~EUSCI_B_CTLW0_SWRST;  // Initialize USCI state machine
}
```

EEPROM.h

```
#include <msp.h>
#include <stdint.h>

uint16_t TransmitFlag = 0;

/*
/ Initialize I2C bus for communicating via I2C.
*/
void InitI2C(uint8_t DeviceAddress)
{
    P3->SEL0 |= BIT6 | BIT7; // Set I2C pins of eUSCI_B2
    // Enable eUSCIB0 interrupt in NVIC module
    NVIC->ISER[0] = 1 << ((EUSCIB2_IRQn) & 31);
    // Configure USCI_B0 for I2C mode
    EUSCI_B2->CTLW0 |= EUSCI_A_CTLW0_SWRST; // Software reset enabled
    EUSCI_B2->CTLW0 = EUSCI_A_CTLW0_SWRST | // Remain eUSCI in reset mode
                      EUSCI_B_CTLW0_MODE_3 | // I2C mode
                      EUSCI_B_CTLW0_MST | // Master mode
                      EUSCI_B_CTLW0_SYNC | // Sync mode
                      EUSCI_B_CTLW0_SSEL__SMCLK; // SMCLK
    EUSCI_B2->BRW = 30; // baudrate = SMCLK / 30 = 100kHz
    EUSCI_B2->I2CSA = DeviceAddress; // Slave address
    EUSCI_B2->CTLW0 &= ~EUSCI_A_CTLW0_SWRST; // Release eUSCI from reset
    EUSCI_B2->IE |= EUSCI_A_IE_RXIE | // Enable receive interrupt
                    EUSCI_A_IE_TXIE;
}



// Function that writes a single byte to the Temp module.

void WriteTemp(uint8_t MemAddress, uint16_t MemByte)
{

    uint8_t HiCtl;
    uint8_t LoCtl;
    HiCtl = MemByte >> 8;
    LoCtl = MemByte & 0xFF;
    EUSCI_B2->CTLW0 |= EUSCI_B_CTLW0_TR; // Set transmit mode (write)
    EUSCI_B2->CTLW0 |= EUSCI_B_CTLW0_TXSTT; // I2C start condition
    while (!TransmitFlag); // Wait for EEPROM address to transmit
    TransmitFlag = 0;
    EUSCI_B2 -> TXBUF = MemAddress; // Send the high byte of the memory address
    while (!TransmitFlag); // Wait for the transmit to complete
    TransmitFlag = 0;
    EUSCI_B2 -> TXBUF = HiCtl; // Send the byte to store in EEPROM
    while (!TransmitFlag); // Wait for the transmit to complete
    TransmitFlag = 0;
    EUSCI_B2 -> TXBUF = LoCtl; // Send the byte to store in EEPROM
```

```
    while (!TransmitFlag); // Wait for the transmit to complete
    TransmitFlag = 0;
    EUSCI_B2 -> CTLW0 |= EUSCI_B_CTLW0_TXSTP; // I2C stop condition
}



// Function that reads a single byte from the Temp module.

uint16_t ReadTemp(uint8_t MemAddress)
{
 uint8_t ReceiveByte1;
 uint8_t ReceiveByte2;
 uint16_t temp;

 EUSCI_B2->CTLW0 |= EUSCI_B_CTLW0_TR; // Set transmit mode (write)
 EUSCI_B2->CTLW0 |= EUSCI_B_CTLW0_TXSTT; // I2C start condition
 while (!TransmitFlag); // Wait for EEPROM address to transmit
 TransmitFlag = 0;
 EUSCI_B2 -> TXBUF = MemAddress; // Send the high byte of the memory address
 while (!TransmitFlag); // Wait for the transmit to complete
 TransmitFlag = 0;

 EUSCI_B2->CTLW0 &= ~EUSCI_B_CTLW0_TR; // Set receive mode (read)
 EUSCI_B2->CTLW0 |= EUSCI_B_CTLW0_TXSTT; // I2C start condition (restart)
 // Wait for start to be transmitted
 while ((EUSCI_B2->CTLW0 & EUSCI_B_CTLW0_TXSTT));

 while (!TransmitFlag); // Wait to receive a byte
 TransmitFlag = 0;
 ReceiveByte1 = EUSCI_B2->RXBUF; // Read byte from the buffer

 // set stop bit to trigger after first byte
 EUSCI_B2->CTLW0 |= EUSCI_B_CTLW0_TXSTP;

 while (!TransmitFlag); // Wait to receive a byte
 TransmitFlag = 0;
 ReceiveByte2 = EUSCI_B2->RXBUF; // Read byte from the buffer

 temp= (ReceiveByte1<<8)+ReceiveByte2;
 return(temp);

}


// Function that writes a single byte to the EEPROM.

void WriteEEPROM(uint16_t MemAddress, uint8_t MemByte)
{
    uint8_t HiAddress;
    uint8_t LoAddress;
```

```
    HiAddress = MemAddress >> 8;
    LoAddress = MemAddress & 0xFF;
    EUSCI_B2->CTLW0 |= EUSCI_B_CTLW0_TR; // Set transmit mode (write)
    EUSCI_B2->CTLW0 |= EUSCI_B_CTLW0_TXSTT; // I2C start condition
    while (!TransmitFlag); // Wait for EEPROM address to transmit
    TransmitFlag = 0;
    EUSCI_B2 -> TXBUF = HiAddress; // Send the high byte of the memory address
    while (!TransmitFlag); // Wait for the transmit to complete
    TransmitFlag = 0;
    EUSCI_B2 -> TXBUF = LoAddress; // Send the low byte of the memory address
    while (!TransmitFlag); // Wait for the transmit to complete
    TransmitFlag = 0;
    EUSCI_B2 -> TXBUF = MemByte; // Send the byte to store in EEPROM
    while (!TransmitFlag); // Wait for the transmit to complete
    TransmitFlag = 0;
    EUSCI_B2 -> CTLW0 |= EUSCI_B_CTLW0_TXSTP; // I2C stop condition
}



// Function that reads a single byte from the EEPROM.

uint8_t ReadEEPROM(uint16_t MemAddress)
{
 uint8_t ReceiveByte;
 uint8_t HiAddress;
 uint8_t LoAddress;
 HiAddress = MemAddress >> 8;
 LoAddress = MemAddress & 0xFF;
 EUSCI_B2->CTLW0 |= EUSCI_B_CTLW0_TR; // Set transmit mode (write)
 EUSCI_B2->CTLW0 |= EUSCI_B_CTLW0_TXSTT; // I2C start condition
 while (!TransmitFlag); // Wait for EEPROM address to transmit
 TransmitFlag = 0;
 EUSCI_B2 -> TXBUF = HiAddress; // Send the high byte of the memory address
 while (!TransmitFlag); // Wait for the transmit to complete
 TransmitFlag = 0;
 EUSCI_B2 -> TXBUF = LoAddress; // Send the low byte of the memory address
 while (!TransmitFlag); // Wait for the transmit to complete
 TransmitFlag = 0;
 EUSCI_B2->CTLW0 &= ~EUSCI_B_CTLW0_TR; // Set receive mode (read)
 EUSCI_B2->CTLW0 |= EUSCI_B_CTLW0_TXSTT; // I2C start condition (restart)
 // Wait for start to be transmitted
 while ((EUSCI_B2->CTLW0 & EUSCI_B_CTLW0_TXSTT));
 // set stop bit to trigger after first byte
 EUSCI_B2->CTLW0 |= EUSCI_B_CTLW0_TXSTP;
 while (!TransmitFlag); // Wait to receive a byte
 TransmitFlag = 0;
 ReceiveByte = EUSCI_B2->RXBUF; // Read byte from the buffer
 return ReceiveByte;
}
```

```
/*
/ I2C Interrupt Service Routine
*/
void EUSCIB2_IRQHandler(void)
{
 if (EUSCI_B2->IFG & EUSCI_B_IFG_TXIFG0) // Check if transmit complete
 {
 EUSCI_B2->IFG &= ~ EUSCI_B_IFG_TXIFG0; // Clear interrupt flag
 TransmitFlag = 1; // Set global flag
 }
 if (EUSCI_B2->IFG & EUSCI_B_IFG_RXIFG0) // Check if receive complete
 {
 EUSCI_B2->IFG &= ~ EUSCI_B_IFG_RXIFG0; // Clear interrupt flag
 TransmitFlag = 1; // Set global flag
 }
}
```

DAC.h

```
#define GAIN BIT5
#define SHDN BIT4
#define DAC_CS1  BIT0

void DAC1(uint16_t data){
    uint8_t hiByte, loByte;

          loByte = 0xFF & data;         // mask just low 8 bits
          hiByte = 0x0F & (data >> 8);  // shift and mask bits for D11-D8
          hiByte |= (GAIN | SHDN);      // set the gain / shutdown control bits

          P1->OUT &= ~DAC_CS1; // Set CS low

          // wait for TXBUF to be empty before writing high byte
          while(!(EUSCI_B0->IFG & EUSCI_B_IFG_TXIFG));
          EUSCI_B0->TXBUF = hiByte;

          // wait for TXBUF to be empty before writing low byte
          while(!(EUSCI_B0->IFG & EUSCI_B_IFG_TXIFG));
          EUSCI_B0->TXBUF = loByte;

          // wait for RXBUF to be empty before changing CS
          while(!(EUSCI_B0->IFG & EUSCI_B_IFG_RXIFG));

          P1->OUT |= DAC_CS1; // Set CS high

}
```

LCD.h

```
#include "msp.h"
#include <stdint.h>
#include "string.h"
#define RS 0x01 //5.0
#define RW 0x02 //5.1
#define EN 0x04 //5.2

void LCD_command(unsigned char command) {
    P5->OUT &= ~(RS | RW);          //RS = 0, R/W = 0
    P4->OUT = command;              //put command on a data bus
    P5->OUT |= EN;                  //pulse E high
    delay_ms(0, 3000000);
    P5->OUT &= ~EN;                 //clear E
    if (command < 4)                //command 1 and 2 need up to 1.64 ms
        delay_ms(4, 3000000);
    else                            //all others 40 us
        delay_ms(1, 3000000);
}


void LCD_init(void){
    P5->DIR |= RS | RW | EN;        //readies outputs
    P4->DIR = 0xFF;

    delay_ms(30, 3000000);          //initialization sequence
    LCD_command(0x30);
    delay_ms(10, 3000000);
    LCD_command(0x30);
    delay_ms(1, 3000000);
    LCD_command(0x30);

    LCD_command(0x38);              //set 8-bit data, 2-line, 5x7 font
    LCD_command(0x06);              //move cursor right after each char
    LCD_command(0x01);              //clear screen, move cursor home
    LCD_command(0x0F);              //turn on display, cursor blinking
}

void clear_LCD(void){
    P5->OUT &= ~(RS | RW);          //RS= 0, R/W = 0
    P4->OUT = 0x01;                 //command 1 (clear)
    P5->OUT |= EN;                  //pulse E high
    delay_ms(0, 3000000);
    P5->OUT &= ~EN;                 //clear E
    delay_ms(4, 3000000);           //command 1 needs up to 1.64 ms
}

void home_LCD(void){
    P5->OUT &= ~(RS | RW);          //RS= 0, R/W = 0
    P4->OUT = 0x02;                 //command 2 (home)
```

```
    P5->OUT |= EN;                  //pulse E high
    delay_ms(0, 3000000);
    P5->OUT &= ~EN;                 //clear E
    delay_ms(4, 3000000);           //command 2 needs up to 1.64 ms
}
void write_num_LCD(int key){
    unsigned char data;
    if (key==0)
        data='0';
    else if (key==1)
        data='1';
    else if (key==2)
        data='2';
    else if (key==3)
        data='3';
    else if (key==4)
        data='4';
    else if (key==5)
        data='5';
    else if (key==6)
        data='6';
    else if (key==7)
        data='7';
    else if (key==8)
        data='8';
    else if (key==9)
        data='9';

    P5->OUT |= RS;                  //RS = 1
    P5->OUT &= ~RW;                 //R/W = 0
    P4->OUT = data;                 //put data on bus
    P5->OUT |= EN;                  //pulse E
    delay_ms(0, 3000000);
    P5->OUT &= ~EN;                 //clear E
    delay_ms(1, 3000000);           //wait for controller to display
}

void write_char_LCD(unsigned char data){
    P5->OUT |= RS;                  //RS = 1
    P5->OUT &= ~RW;                 //R/W = 0
    P4->OUT = data;                 //put data on bus
    P5->OUT |= EN;                  //pulse E
    delay_ms(0, 3000000);
    P5->OUT &= ~EN;                 //clear E
    delay_ms(1, 3000000);           //wait for controller to display
}

void first_line_LCD(void){
    P5->OUT &= ~(RS | RW);          //RS= 0, R/W = 0
    P4->OUT = 0x80;                 //command 80 (cursor first line)
    P5->OUT |= EN;                  //pulse E high
```

```
    delay_ms(0, 3000000);
    P5->OUT &= ~EN;                 //clear E
    delay_ms(1, 3000000);           //commands need 40 us
}

void second_line_LCD(void){
    P5->OUT &= ~(RS | RW);          //RS= 0, R/W = 0
    P4->OUT = 0xC0;                 //command C0 (cursor second line)
    P5->OUT |= EN;                  //pulse E high
    delay_ms(0, 3000000);
    P5->OUT &= ~EN;                 //clear E
    delay_ms(1, 3000000);           //ommands need 40 us
}
```

Keypad.h

```
#define COL1  BIT4
#define COL2  BIT5
#define COL3  BIT6
#define ROW1  BIT0
#define ROW2  BIT1
#define ROW3  BIT2
#define ROW4  BIT3

/* this function initializes Port 2 that is connected to the keypad.
 * All pins are configured as GPIO input pin. The row pins have
 * the pull-down resistors enabled.
 */
void keypad_init(void) {
    P2->DIR = 0;        // make all pins an input
    P2->REN |= (ROW1 | ROW2 | ROW3 | ROW4);  // enable resistor for row pins
    P2->OUT &= ~(ROW1 | ROW2 | ROW3 | ROW4); // make row pins pull-down
}


uint8_t keypad_getkey(void) {
    uint8_t row, col, key;

    /* check to see any key pressed */
    P2->DIR |= (COL1 | COL2 | COL3);  // make the column pins outputs
    P2->OUT |= (COL1 | COL2 | COL3);  // drive all column pins high
    delay_ms(1,3000000);                // wait for signals to settle

    row = P2->IN & (ROW1 | ROW2 | ROW3 | ROW4);   // read all row pins

    if (row == 0)           // if all rows are low, no key pressed
        return 0xFF;

    /* If a key is pressed, it gets here to find out which key.
     * It activates one column at a time and reads the input to see
     * which row is active. */

    for (col = 0; col < 3; col++) {
        // zero out bits 6-4
        P2->OUT &= ~(COL1 | COL2 | COL3);

        // shift a 1 into the correct column depending on which to turn on
        P2->OUT |= (COL1 << col);
        delay_ms(1,3000000);            // wait for signals to settle

        row = P2->IN & (ROW1 | ROW2 | ROW3 | ROW4); // mask only the row pins

        if (row != 0) break;      // if the input is non-zero, key detected
    }
```

```
    P2->OUT &= ~(COL1 | COL2 | COL3);   // drive all columns low
    P2->DIR &= ~(COL1 | COL2 | COL3);   // disable the column outputs

    if (col == 3)   return 0xFF;        // if we get here, no key was detected

    // rows are read in binary, so powers of 2 (1,2,4,8)
    if (row == 4) row = 3;
    if (row == 8) row = 4;

    /*****************************************************************
     * IF MULTIPLE KEYS IN A COLUMN ARE PRESSED THIS WILL BE INCORRECT *
     *****************************************************************/

    // calculate the key value based on the row and columns where detected
    if (col == 0) key = row*3 - 2;
    if (col == 1) key = row*3 - 1;
    if (col == 2) key = row*3;

    if (key == 11)  key = 0; // fix for 0 key

    return key;
}
```

delay_ms.h

```
void delay_ms(int ms, int freq) {
    int i, j;
    for (j=0; j<ms; j++)
    for (i=freq/10000; i>0; i--);
}

void delay_us(int us, int freq) {
    int i, j;
    for (j=0; j<us; j++)
    for (i=freq/10000000; i>0; i--);
}
```